VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

DIPLOMOVÁ PRÁCE

Brno, 2016 Bc. Petr Máchal

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ
DEPARTMENT OF TELECOMMUNICATIONS

NEPROFILUJÍCÍ ÚTOKY PROUDOVOU ANALÝZOU
NON-PROFILING POWER ANALYSIS ATTACKS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. Petr Máchal

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. Zdeněk Martinásek, Ph.D.

BRNO 2016

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní obor **Telekomunikační a informační technika**
Ústav telekomunikací

Student: Bc. Petr Máchal ***ID:*** 14398
Ročník: 2 ***Akademický rok:*** 2015/16

NÁZEV TÉMATU:

Neprofilující útoky proudovou analýzou

POKYNY PRO VYPRACOVÁNÍ:

V rámci diplomové práce prostudujte možnosti prolomení šifrovacího algoritmu AES pomocí proudové analýzy. V úvodu práce vytvořte přehled současného stavu problematiky (zaměřte se jen na neprofilující útoky DPA). Cílem práce je využít proudových průběhů soutěže DPA Contest 4.2 http://www.dpacontest.org a implementovat různé neprofilující útoky využívající různé statistické nástroje k určení závislosti odhadů proudové spotřeby a skutečné proudové spotřeby (korelační koeficient, vzájemná informace atd.). Cílem práce je porovnat tyto metody z pohledu efektivity útoku (využijte metriku PGE na počtu proudových průběhů). Zvolené metody DPA implementujte v prostředí Matlab.

DOPORUČENÁ LITERATURA:

[1] MANGARD, Stefan a OSWALD, Elisabeth a POPP, Thomas: Power Analysis Attacks: Revealing the Secrets of Smart Cards (Advances in Information Security). Secaucus, NJ, USA:Springer-Verlag New York, Inc., 2007, ISBN 0387308571.

[2] KOCHER, Paul a JAFFE, Joshua.; JUN Benjamin: Differential Power Analysis. In CRYPTO '99: Proceedings of the 19th Annual International Cryptology Conference onAdvances in Cryptology, London, UK: Springer-Verlag, 1999, ISBN 3-540-66347-9, s. 388–397.

Termín zadání: 1.2.2016 ***Termín odevzdání:*** 25.5.2016

Vedoucí práce: Ing. Zdeněk Martinásek, Ph.D.
Konzultant diplomové práce:

doc. Ing. Jiří Mišurec, CSc., *předseda oborové rady*

ABSTRAKT

Tato práce se zabývá možnostmi prolomení šifrovacího algoritmu AES pomocí neprofilujících útoků diferenciální proudovou analýzou. V úvodu jsou uvedeny v současnosti používané techniky diferenciální analýzy a pro přehled je zmíněna i jednoduchá proudová analýza. V dalších kapitolách jsou stručně popsány ochrany před útoky a popsán šifrovací algoritmus AES. Nejdůležitější část tvoří kapitoly s popisem realizace útoků na algoritmus AES-128 pomocí korelační analýzy a analýzy vzájemné informace. Tyto útoky využívají proudové průběhy ze stránek věnovaných knize Power Analysis Attacks - Revealing the Secrets of Smartcards, http://DPAbook.org a především pak proudovým průběhům ze soutěže DPA Contest 4.2, http://www.dpacontest.org. Na závěr je srovnání metod na základě počtu proudových průběhů nutných k nalezení klíče zprávy.

KLÍČOVÁ SLOVA

Proudová analýza, neprofilující útok, diferenciální proudová analýza, proudový postranní kanál, analýza vzájemnou informací

ABSTRACT

The work is mainly concerned with the possibilities of breaking the encryption algorithm AES with using of non-template attacks. In the introduction are listed techniques of differential analysis, which are using in the present, but for the sake of completeness is there mention about simple power analysis. In the next chapters are briefly described countermeasures against power analysis and further is described the AES algorithm. Most important parts are chapters where are described attack implementation on AES-128 through correlation power analysis and mutual information analysis. These attacks exploit power traces from www pages dedicated to book Power Analysis Attacks - Revealing the Secrets of Smartcards, http://DPAbook.org and especially to power traces from DPA Contest 4.2, http://www.dpacontest.org. In conclusion is comparison of methods based on the number of power traces needed for finding the key of secret message.

KEYWORDS

Power analysis, non-template based attack, differential power analysis, power side channel, mutual information analysis

MÁCHAL, Petr *Neprofilující útoky proudovou analýzou*: diplomová práce. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2016. 65 s. Vedoucí práce byl Ing. Zdeněk Martinásek, Ph.D.

PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma „Neprofilující útoky proudovou analýzou“ jsem vypracoval(a) samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor(ka) uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil(a) autorská práva třetích osob, zejména jsem nezasáhl(a) nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom(a) následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

podpis autora

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu diplomové práce panu Ing. Zdeňku Martináskovi, Ph.D. za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

Brno ....

podpis autora

Faculty of Electrical Engineering
and Communication
Brno University of Technology
Purkynova 118, CZ-61200 Brno
Czech Republic

http://www.six.feec.vutbr.cz

PODĚKOVÁNÍ

Výzkum popsaný v této diplomové práci byl realizován v laboratořích podpořených projektem Centrum senzorických, informačních a komunikačních systmémů (SIX); registrační číslo CZ.1.05/2.1.00/03.0072, operačního programu Výzkum a vývoj pro inovace.

EVROPSKÁ UNIE
EVROPSKÝ FOND PRO REGIONÁLNÍ ROZVOJ
INVESTICE DO VAŠÍ BUDOUCNOSTI

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

1 ÚVOD

Útoky na moderní šifrovací algoritmy jsou často velmi náročné. Algoritmy jsou navrhovány tak, aby jejich prolomení bylo časově náročné. Běžně je proto možné se setkat s algoritmy u nichž by prolomení klíče běžnými metodami na běžném počítači znamenalo čas delší než je stáří našeho vesmíru. Do popředí se tak dostávají techniky, které umožňují časovou náročnost zkrátit, pokud je to možné.

Jedním ze způsobů jak se pokusit prolomit šifrovací algoritmus je útok takzvaným postranním kanálem. Jedná se pasivní útok, který se nepokouší využít slabosti šifrovacího algoritmu, ale útočí na slabá místa kryptografického algoritmu. To může mít za následek únik informací, které pak mohou být útočníkem zneužity. Pokud je totiž uniklá informace nějakým způsobem závislá na tajném klíči, je možné se pokusit tento tajný klíč rekonstruovat. Postranní kanál je tedy jakákoliv nežádoucí výměna informací mezi šifrovacím systémem a jeho okolím. Postranní kanály můžeme do určité míry minimalizovat. Obecně můžeme ale říci, že všechny kryptografické systémy jsou zranitelné útokem postranním kanálem. Momentálně neexistuje žádný návrh kryptografického zařízení, který by byl zcela imunní vůči všem druhům postranních kanálů.

První práce o útoku postranním kanálem byla publikována v roce 1995 [8]. Jednalo se o útok pomocí časové analýzy, kde Paul Kocher popsal způsob jak kompromitovat klíče RSA, DSS a dalších šifrovacích systémů měřením doby provádění šifrovacího algoritmu. V roce 1998 byla Paulem Kocherem a vědci okolo něj představena ještě mnohem účinnější metoda útoku postranním kanálem, takzvaná Proudová analýza. Během procesu šifrování jsou zachycovány hodnoty spotřebovávaného proudu, které jsou potom zpracovány pomocí Jednoduché proudové analýzy (SPA) nebo Diferenciální proudové analýzy (DPA). V roce 2001 bylo pro útok postranním kanálem využito elektromagnetické záření, viz [9] a [10]. Tento způsob útoku přináší mnoho různých informací, přičemž může být prováděn i na určitou vzdálenost. Další typy postraních kanálů jsou uvedeny v práci [6].

Tato diplomová práce se zabývá útoky na algoritmus AES, vedené diferenciální proudovou analýzou a některými dalšími druhy analýz, které jsou z ní odvozené. V práci je zmíněna technika SPA jako příklad profilujícího útoku a ve zkratce popsána DPA. Budou ukázány tři proudové modely, mezi nimiž je i model Hammingovy váhy, který je často používán a je použit i v praktické části této práce. Hlavním úkolem této práce je použití různých statistických metod k odhalení tajného klíče a jejich porovnání, proto jsou některé popsány v dalších kapitolách.

2 ÚTOKY PROUDOVÝM POSTRANNÍM KANÁLEM

Výkonová analýza studuje závislost proudové spotřeby šifrovacího zařízení na jeho činnosti. Tato práce se nezabývá okolnostmi proč tomu tak je. Souvislosti mezi spotřebou a ději, kte kterým dochází v šifrovacích obvodech jsou dostatečně vysvětleny v pracích [1], [2], [6]. Proto přejdeme přímo ke stručným charakteristikám útoků.

2.1 Jednoduchá proudová analýza

Jednoduchá proudová analýza byla popsána Kocherem [1] jako technika, která přímo interpretuje naměřené hodnoty proudové spotřeby šifrovacího systému získané během šifrování. Často vyžaduje detailní znalosti o způsobu implementace šifrovacího algoritmu použitém v zařízení, které je cílem útoku.

Útok pomocí SPA je užitečný ve chvíli, kdy je znám pouze jeden nebo jen několik průběhů proudové spotřeby. Samozřejmě je lépe mít více průběhů, protože je můžeme využít k potlačení šumu výpočtem průměrné hodnoty průběhu. Množství změřených průběhů nic nemění na principu SPA. Útočník musí být vždy schopen měřit proudovou spotřebu při šifrování a zároveň musí mít šifrování přímý nebo nepřímý dopad na proudovou spotřebu.

2.1.1 Útok pomocí šablon

Jedním z druhů SPA útoků je útok pomocí šablon neboli profilů. Tento útok využívá toho, že spotřeba závisí na zpracovávaných datech. Proudové průběhy jsou charakterizovány vícerozměrným normálním rozdělením, které je plně definováno vektorem středních hodnot a kovarianční maticí (**m;C**). Této dvojici říkáme *šablona*. Oproti ostatním útokům můžeme profilující útok rozdělit na dvě fáze a to na fázi získávání charakteristických dat a na fázi, kdy jsou získaná data použita k útoku. Příkladem může být získání klíče z šifrovací karty, pokud útočník vlastní stejný typ karty. Nejprve spustí na této kartě sekvenci instrukcí pro různá vstupní data d_i a pro různé šifrovací klíče k_j. Získané proudové průběhy si zaznamená. Po té seskupí odpovídající průběhy (d_i,k_j) a vypočte vektor středních hodnot a kovarianční matici vícerozměrného normálního rozdělení. Výsledkem toho jsou šablony pro všechny dvojice dat a klíčů (d_i,k_j) : $h_{d_i,k_j} = (\mathbf{m;C})_{d_i,k_j}$. Získaná data pak použije k útoku na cizí kartu. Vypočte pravděpodobnosti pro změřenou proudovou spotřebu pro všechny šablony

podle vztahu:

$$p(t;(m;C)_{d_i,k_j}) = \frac{\exp(\frac{1}{2}\cdot(t-m)'\cdot C^{-1}\cdot(t-m))}{\sqrt{(2\cdot\pi)^T\cdot det(C)}} \tag{2.1}$$

Jak je dáno zadáním, tato práce se nemá zbývat profilujícími útoky ani SPA, proto pro více informací doporučuji vyhledat [1], [2], [6].

2.2 Diferenciální proudová analýza

Diferenciální proudová analýza je nejpopulárnějším druhem útoku proudovou analýzou. Je to způsobeno faktem, že na rozdíl od SPA není potřeba mít mnoho znalostí o šifrovacím systému, na který je útočeno. Zároveň je možné ji použít i na silně zašuměná data získaná v průběhu útoku. Dostačující znalostí pro použití DPA je už to, že víme jaký šifrovací algoritmus je používán.

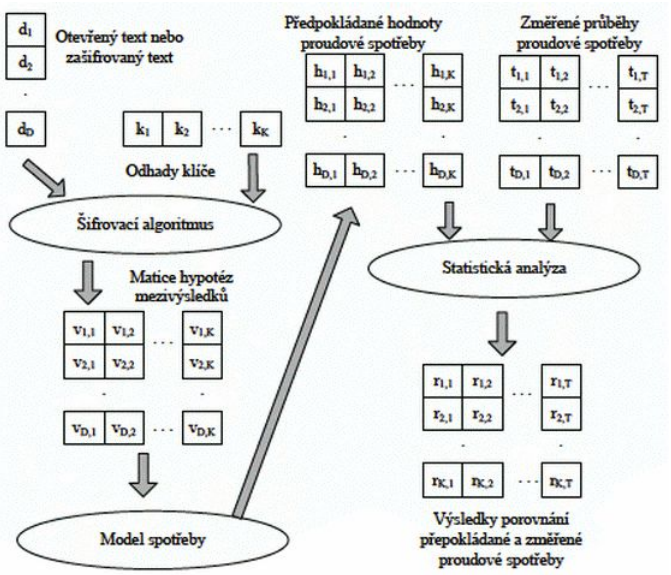
Na rozdíl od SPA ale vyžaduje velké množství změřených průběhů ať už z operace šifrování nebo z dešifrování. Nezáleží tedy na tom, jestli jako vstupní data použijeme prostý nebo zašifrovaný text. Dalším rozdílem je přístup ke zpracovaným datům. Zatímco SPA se snaží v průbězích najít nějakou stopu, otisk, vzor tak v DPA není tvar průběhu až tak důležitý. DPA analyzuje jak v přesně daném časovém okamžiku závisí proudová spotřeba na přenášených datech.

DPA má přesně určený postup útoku. Lze jej rozdělit do pěti kroků. Viz Obrázek 2.1.

1. **Volba mezivýsledku algoritmu.**
2. **Měření proudové spotřeby**
3. **Výpočet hypotetických mezivýsledků**
4. **Mapování mezivýsledků na hodnoty proudové spotřeby**
5. **Porovnání hypotetických hodnot s naměřenými proudovými průběhy**

Všechny výše uvedené kroky jsou dostatečně popsány v [2], [6].

Toto schéma je ale možné použít i na další typy proudových analýz. První tři body zůstávají stejné, k rozdílu dochází až v bodu 4, kde můžeme zvolit proudový model, nebo v bodu 5 kde se volí typ statistické analýzy. Je populární vytvářet "nové"typy proudových analýz. Často ale jde ve výsledku právě pouze o použití jiné statistické metody v pátém kroku DPA.



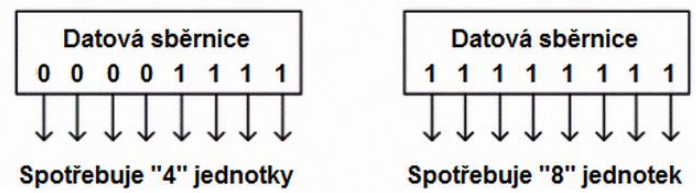
Obr. 2.1: Schéma 3. až 5. kroku útoku DPA

2.3 Proudové modely

K výběru proudového modelu dochází ve čtvrtém kroku ze schematu na Obrázku 2.1. Nejznámějšími a nejpoužívanějšími jsou model Hammingovy váhy a Hammingovy vzdálenosti. Dalším používaným modelem je model Nulové hodnoty. Výběr vhodného proudového modelu závisí na zařízení na které je útočeno. Model Hammingovy váhy se používá nejčastěji na čipové karty, Hammingovy vzdálenosti na mikrokontroléry, model Nulové hodnoty na hardwarové implementace. Případně je potřeba vyzkoušet, který z nich je nejvhodnější.

2.3.1 Hammingova váha

Model Hammingovy váhy (HW) je základním modelem proudové spotřeby. Využívá předpokladu, že spotřeba proudu obvodu je přímo úměrně závislá na počtu nenulových bitů. Viz Obrázek 2.2.



Obr. 2.2: Příklad Hammingovy váhy

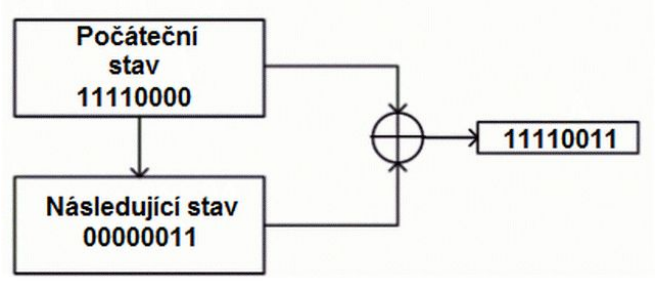
Tento model je možné použít i v případě, kdy nemáme dostatek informací o cílovém obvodu. V porovnání s modelem Hammingovy vzdálenosti není tak efektivní.

2.3.2 Hammingova vzdálenost

Model Hammingovy vzdálenosti (HD) je rozšířením modelu Hammingovy váhy. K určení proudové spotřeby využívá sledování změn logických hodnot v daném časovém intervalu. Ke změnám může docházet v mnoha částech obvodu jako jsou například sběrnice, registry, paměť. Proudová spotřeba je určena na základě přechodů $0 \rightarrow 1$ a $1 \rightarrow 0$. Počet bitových přechodů je Hammingova váha XOR hodnoty mezi dvěma hodnotami. Například pro dvě v registru uložené hodnoty R0 a R1 je HD:

$$HD(R0, R1) = HW(R0 \oplus R1). \tag{2.2}$$

Při použití tohoto modelu se vychází z několika předpokladů. Předpokládá se, že bity které se nemění $0 \rightarrow 0$ a $1 \rightarrow 1$ nepřispívají ke spotřebě obvodu. Také se předpokládá, že změna $0 \rightarrow 1$ a $1 \rightarrow 0$ spotřebuje stejné množství proudu.



Obr. 2.3: Příklad Hammingovy vzdálenosti

Na Obrázku 2.3 je příklad výpočtu HD. Hodnota HD přechodu z počátečního stavu do dalšího odpovídá počtu jedniček ve výsledku, tedy HD = 6.

2.3.3 Model nulové hodnoty

DPA útoky založené na HD modelu jsou docela úspěšné. Je to způsobeno tím, že HD model docela dobře popisuje spotřebu energie AES čipu v okamžiku, kdy je šifrovaný text uložen do registru, který obsahuje vstupy S-boxu. Nicméně je možné využít i spotřebu energie, která je způsobena výpočtem S-boxů v poslední rundě.

S-boxy mohou být implementovány třemi různými způsoby. Mohou být vytvořeny v paměti ROM, mohou být vytvořeny jako vyhledávací tabulka, nebo pomocí výpočtu převrácených hodnot a afinních zobrazení. Poslední varianta má tu vlastnost, že pokud je hodnota na vstupu S-boxu nula, tak spotřebovává významně méně energie než ve všech ostatních případech. To je možné vysvětlit tím, že neexistuje

převrácená hodnota nuly, stejně tak je při násobení nulou výsledkem zase nula. Model nulové hodnoty můžeme zapsat jako:

$$h_{i,j} = ZV(v_{i,j}) = \begin{cases} 0 & pro\ v_{i,j} = 0 \\ 1 & pro\ v_{i,j} \neq 0 \end{cases} \tag{2.3}$$

ZV model použijeme tedy tak, že pro každou nulovou hypotetickou hodnotu S-boxu nastavíme hosnotu hypotetické proudové spotřeby také na nulu. Více informací lze najít v [2] a [3].

2.4 Používané typy statistických analýz

Původně (viz. [1]) byla DPA navržena a aplikována na dešifrování algoritmu DES pomocí metody založené na rozdílu středních hodnot. V této práci budou předvedeny další statistické analýzy, jako jsou Korelační proudová analýza, Analýza vzájemnou informací, použití Kolmogoro-Smirnovova a Cramér–von Misesova testu.

2.4.1 Korelační proudová analýza

Jak bylo řečeno v kapitole 1.2 jsou další některé proudové analýzy pouze užitím jiné statistické metody. Korelační proudová analýza (CPA) je přesně tento případ. Místo rozdílu středních hodnot používá jako statistickou metodu v pátém kroku schématu z Obrázku 2.1 výpočet korelačního koeficientu.

Korelační koeficient popisuje lineární vztah mezi dvěma proměnnými. Pro veličiny X a Y je definován jako pomocí kovariance dle vztahu:

$$\rho(X,Y) = \frac{Cov(X,Y)}{\sqrt{\sigma^2(X) \cdot \sigma^2(X)}} \tag{2.4}$$

Korelační koeficient vždy leží v intervalu $\langle -1; 1 \rangle$. Pokud je roven -1, jsou na sobě proměnné nepřímo závislé. Pokud je roven 1, jsou na sobě proměnné přímo závislé. Korelační koeficient rovný 0 říká, že mezi proměnnými není žádná zjistitelná lineární závislost.

V DPA se korelační koeficient používá k určení závislosti mezi sloupci předpokládaných h_i a naměřených t_j hodnot, pro *i=1,...,K* a *i=1,...,T*. Výsledkem je matice **R** obsahující korelační koeficienty. Jednotlivé koeficienty lze získat ze vztahu:

$$r_{i,j} = \frac{\sum_{d=1}^{D}(h_{(d,i)} - \bar{h}_i) \cdot (t_{(d,j)} - \bar{t}_j)}{\sqrt{\sum_{d=1}^{D}(h_{(d,i)} - \bar{h}_i)^2 \cdot (t_{(d,j)} - \bar{t}_j)^2}} \tag{2.5}$$

kde $\bar{h}_i \bar{t}_j$ označují průměrné hodnoty sloupců h_i a t_j.

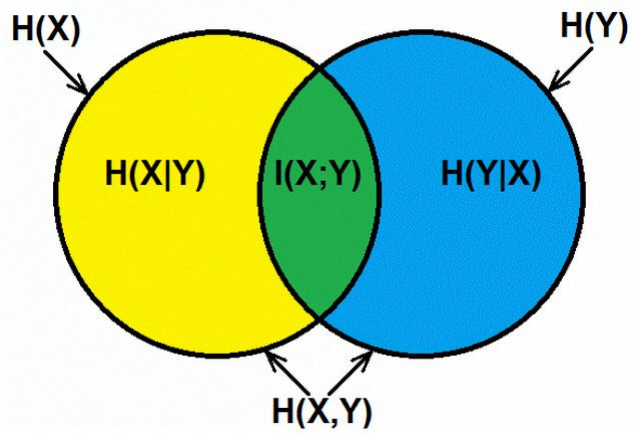
2.4.2 Vzájemná informace

Vzájemná informace dvou proměnných je míra množství informace, kterou každá z nich nese o té druhé.

Vzájemnou informaci dvou diskrétních náhodných proměnných X a Y lze zapsat jako:

$$I(X,Y) = \sum_{y \in Y} \sum_{x \in X} p(x,y) \log \left(p \frac{(x,y)}{p(x)p(y)} \right) \tag{2.6}$$

kde *p(x,y)* je sdružená hustota pravděpodobnosti X a Y a *p(x)* resp. *p(y)* jsou okrajové distribuční funkce proměnných X resp. Y.



Obr. 2.4: Vztahy mezi entropiemi

Na Obrázku 2.4 jsou pomocí Vennova diagramu zobrazeny vztahy mezi entropiemi H(X) a H(Y), sdruženou entropií H(X,Y), podmíněnou entropií H(X|Y), H(Y|X) a vzájemnou informací I(X; Y) pro dvojici náhodných proměnných X, Y. Vyplývají z něho dva extrémy. Pokud budou X a Y dvě nezávisle proměnné, tak X nebude říkat nic o Y, entropie H(X) a H(Y) nebudou mít žádný průnik a hodnota vzájemné informace tak bude nulová. Na druhou stranu pokud bude X funkcí Y, tak potom bude celá informace nesená X obsažena v Y a hodnota vzájemné informace bude maximální. Vzájemná informace tedy nabývá hodnot nula a větších.

Vyjádřit můžeme vzájemnou informaci také jako:

$$\begin{aligned} I(X;Y) &= H(X) - H(X|Y) = \\ &= H(Y) - H(Y|X) = \\ &= H(X,Y) - H(X|Y) - H(Y|X) = \\ &= \mathbf{H(X) + H(Y) - H(X,Y)}, \end{aligned} \tag{2.7}$$

což odpovídá zobrazení podle Obrázku 2.4. Ve skriptu `mutualinfo.m` je pak při výpočtu prakticky použita zvýrazněná část Rovnice 2.7.

Analýza vzájemnou informací

Koncept analýzy vzájemnou informací byl představen Benediktem Gierchlisem [12] v roce 2008. Metoda je schopna efektivně detekovat lineární i nelineární závislosti mezi předpověďmi a pozorováními unikajících informací. Provedením výpočtu pro všechny hodnoty je možné zjistit tajný klíč. Bude jím ta hypotetická hodnota klíče, která bude mít nejvyšší úroveň vzájemné informace.

Předpokládejme, že máme množinu měření proudových spotřeb $T = \{T_1, ..., T_N\}$. {0,...,m} je množina možných hodnot cílových pomocných proměnných. Tyto proměnné jsou výstupem funkce *F(x_i, k')*, kde *k'* je přepokládaný klíč.

Možný způsob útoku probíhá následujícím způsobem [13]:

1. Je vypočtena entropie *H(T)*.
2. Každému předpokládanému klíči *k'* je přiřazeno x_i odpovídající výstupu F:

$$H_j^{k'} = \{x_i | F(x_i, k') = j\} \text{ pro } j \in \{0, ..., m\} \tag{2.8}$$

3. Podobně pro naměřené průběhy vypočítat:

$$G_j^{k'} = \{T_i | x_i \in H_j^{k'}\} \text{ pro } \{0, ..., m\} \tag{2.9}$$

4. Pro každé j vypočítat *H(T/F = j)* použitím $G_j^{k'}$.
5. Vypočítat *H(T/F = j)* pomocí výsledku z předchozího kroku
6. Pro každé *k'* vypočítat:

$$I_{k'}(T, F) = H(T) - H(T|F) \tag{2.10}$$

7. A konečněm určit hodnotu k' jako:

$$I_{k'}(T, F) = \max_k(I_k(T, F)) \tag{2.11}$$

3 OCHRANA PŘED ÚTOKY

Útoky využívající proudový postranní kanál je možné provádět díky tomu, že spotřeba šifrovacího obvodu je závislá na zpracovávaném algoritmu. Cílem protiopatření je tedy zbavit se nebo alespoň velmi omezit tuto závislost a ztížit tak útok postranním proudovým kanálem. Hlavními technikami ochrany jsou skrývání a maskování.

3.1 Skrývání

Skrývání je technika jejíž cílem je změnit spotřebu tak, aby nebyla závislá na hodnotě mezivýsledků odebíraného proudu a na odběru probíhajících operací. Toho je možné docílit dvěma různými způsoby. Buď vytvořit zařízení, které odebírá zcela náhodné množství výkonu, nebo vytvořit zařízení, které bude kdykoliv odebírat stejné množství výkonu.Žádného z těchto cílů ale nejde dosáhnout. Je ale možné se jim přiblížit.

Řešení lze rozdělit do dvou skupin. Buď je možné pro náhodně odebírané množství výkonu navrhnout takový šifrovací algoritmus, který bude stejnou šifrovací operaci provádět v různých časech každého šifrovacího cyklu - změna v časové oblasti, nebo je možné ovlivňovat přímo velikost amplitudy. A to buď náhodnou změnou její velikosti, nebo naopak nastavením její velikosti na konstantní hodnotu.

Ke změnám v časové oblasti se používají:

- Náhodné vkládání falešných operací
- Míchání

V amplitudové oblasti jsou využívány postupy:

- Zvýšení šumu
- Omezení signálu

Více se zde nebudeme technikami skrývání zabývat, protože to není smyslem práce a ani není technika skrývání použita v zadání. Více informací lze nalézt v [2], [6], [14].

3.2 Maskování

Technika maskování přistupuje ke změně závislosti spotřeby jinak. Dosahuje toho přidáním náhodných hodnot nějakým způsobem ke vstupním hodnotám. Výsledkem toho potom je, že hodnoty na výstupu šifrovacího algoritmu nejsou v korelaci se vstupními.

V operaci maskování je každá z hodnot mezivýsledků v ukryta za náhodnou hodnotou m, které se říká maska.

$$v_m = v * m \tag{3.1}$$

Maska m je generována přímo šifrovacím modulem a liší se při každém použití. Proto není známa útočníkovi. Operace $*$ je definována podle toho jaké operace jsou použity šifrovacím algoritmu. Nejčastěji používanou operací je exkluzivní disjunkce (XOR - $\oplus$), modulární sčítání nebo modulární násobení. V případě aritmetických operací je modulo vybráno na základě šifrovacího algoritmu.

Rozeznáváme dva typy maskování, booleanovské a aritmetické. V booleanovském maskování je použita operace XOR, tedy:

$$v_m = v \oplus m. \tag{3.2}$$

V aritmetickém maskování se používá aritmetické sčítání nebo násobení společně s operací modulo:

$$v_m = v + m \pmod{n} \tag{3.3}$$

$$v_m = v \times m \pmod{n} \tag{3.4}$$

Některé algoritmy využívají boolenaovských i aritmetických funkcí. Tento přístup je ale náročnější, neboť vyžaduje významné množství dalších operací. [15] [16]

Šifrovací algoritmy kromě toho využívají lineárních a nelineárních funkcí. Vlastností lineární funkce je, že:

$$f(x * y) = f(x) * f(y). \tag{3.5}$$

Pokud je tedy operací $*$ exkluzivní disjunkce, pak je tedy $f(x \oplus y) = f(x) \oplus f(y)$. V booleanovském maskování je tak jednoduché masky počítat i je měnit. Z toho plyne, že je jednoduché booleanovským maskováním zamaskovat lineární operace.

Operace AES S-box je nelineární operací, protože platí:

$$S(x \oplus y) \neq S(x) \oplus S(y) \tag{3.6}$$

Vzhledem k tomu, že se maska mění složitějším způsobem, je tento případ náročnější na výpočet. Booleanovské maskování proto není jednoduše použitelné. Avšak S-box je založen na výpočtu multiplikativní inverzi konečného tělesa:

$$f(x) = x^{-1}. \tag{3.7}$$

Toto je vhodné pro multiplikativním maskování, protože:

$$f(x \times m) = (x \times m)^{-1} = f(x) \times f(m). \tag{3.8}$$

Multiplikativní maskování má ale jeden zásadní nedostatek. Není pomocí něj možné zamaskovat mezivýsledek s nulovou hodnotou, což je možné využít k útoku na šifrovací algoritmus.

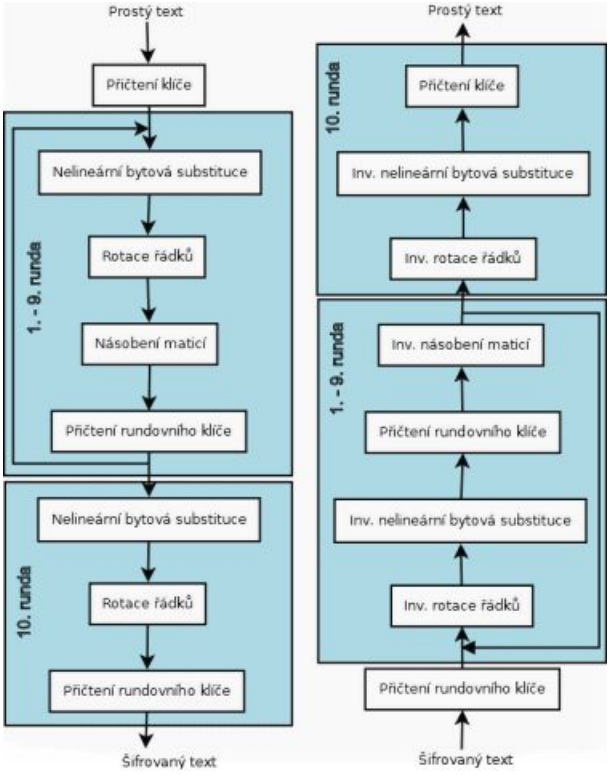
Jak efektivně postupovat v přechodu mezi booleanovským a multiplikativním maskováním je popsáno například v práci [17].

4 ADVANCED ENCRYPTION STANDARD

4.1 Stručný popis

Algoritmus Advanced Encryption Standard (AES) [4], byl navržen pány Rijmenem a Daemenem v roce 1997. V roce 2001 schválil tuto šifru americký úřad pro standardizaci NIST. Je to první široce dostupná šifra, která byla uznána ameriskou Národní Bezpečnostní Agenturou (NSA) pro šifrování dokumentů stupně přísně tajné. Od roku 2002 je používána jako federální standard USA.

AES je standard symetrické blokové šifry, používající pro šifrování i dešifrování stejný klíč. Má pevně danou velikost bloku dat a to 128 bitů, nejčastější velikost klíče je 128 bitů (pak bývá značena jako AES-128). Tato velikost klíče je použita i v této práci. Dalšími možnostmi jsou ještě 192 a 256 bitů. AES pracuje s maticí bytů o velikosti 4x4 bytů, označovanou jako Stav. AES-128 je složena z 10 rund (opakovaných sekvencí blokových operací). Runda obsahují následující operace, viz Obrázek 4.1:



Obr. 4.1: Postup šifrování a dešifrování šifrou AES-128

- Nelineární substituce (SubBytes) - každý byte ze stavu je nahrazen hodnotou z tabulky S-Box, viz Obrázek 4.2. Ta byla navržena tak aby odolala lineární i diferenciální kryptoanalýze.

- Rotace řádků (ShiftRows) - Přesune byty v každém řádku o daný offset (0, 1, 2 a 3 byty)
- Násobení maticí (MixColumns) - Prvky ve stavu jsou vynásobeny pevně danou maticí.
- Přičtení rundovního klíče (AddRoundKey) - Přičtení rundovního klíče operací XOR

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
5	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
6	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
7	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
8	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
9	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
A	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
B	E7	C8	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
C	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
D	70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1D	9E
E	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
F	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

Obr. 4.2: Tabulka S-Box

Pro dešifrování jsou operace provedeny v opačném pořadí, přičemž některé z nich jsou provedeny v opačném pořadí. Více informací o AES viz [6], [4], [2].

Jako cíl útoku je vhodné zvolit některou z nelineárních operací, protože lineární operace ve většině případů nevynášejí dostatečné množství informací. Všechny analýzy se proto zaměřují na nelineární transformaci `SubBytes`.

5 VÝSLEDKY STUDENTSKÉ PRÁCE

5.1 DPAbook.org

V této části je předveden jednodušší skript využívající rozdílu středních hodnot (DPA), korelaci (CPA) a vzájemnou informaci (MIA) pro útok proudovou analýzou. V práci je využita funkce *demo_dpa.m*, která byla doplněna o analýzu vzájemnou informací, a pracovní prostory *WS1.mat* a *WS2.mat* uvedené na stránkách *dpabook.org*. Stránky jsou doplňkem knihy [2]. Výsledkem je komentovaný skript srovnání použitých metod. Funkci pro výpočet vzájemné informace je možné získat ze stránek *mathworks.com*[1].

5.1.1 Popis funkce

Funkce dodržuje postup uvedený v kapitole 2.2 a na Obrázku 2.1. Lze ji tak rozdělit do několika částí.

1. **Volba mezivýsledku algoritmu.**
2. **Měření proudové spotřeby.**
3. **Výpočet hypotetických mezivýsledků.**
4. **Mapování mezivýsledků na hodnoty proudové spotřeby.**
5. **Porovnání hypotetických hodnot s naměřenými proudovými průběhy.**

První dva body jsou realizovány již v daném pracovním prostoru. Proto postačuje zvolený pracovní prostor pouze nahrát do paměti. Na dalším řádku je zvolena hodnota bytu klíče, na který je útočeno.

```
1 load('-mat',workspace);
2 b=1;
```

Ve třetím kroku jsou počítány hypotetické mezivýsledky. Je vygenerována kombinace pro všechny možné klíče a spočítána matice *after_sbox* pro všechny vstupy. Matice má rozměr 200x256. Operace `SubBytes` slouží k převedení hodnot výsledku operace $inputs(i) \oplus key$ pomocí substituční matice, viz Obrázek 4.2.

```
1 [m,n] = size(traces);
2 key = [0:255];
3 after_sbox = zeros(m,256);
```

[1] http://www.mathworks.com/matlabcentral/fileexchange/35625-information-theory-toolbox - autor Mo Chen

```
4
5 for i=1:m
6     after_sbox(i,:) = SubBytes(bitxor(inputs(i),key)+1);
7 end
```

Ve čtvrtém kroku je každé hodnotě z $after\_sbox$ přiřazena hypotetická hodnota proudové spotřeby na základě proudového modelu. V tomto případě Hammingovy váhy. Je samozřejmě možné využít i jiných, ale výsledek není předem zaručen. Pokusil jsem se implementovat model Hammingovy vzdálenosti podle [2]:

$$HD(v_{i,j}, d_i) = HW(v_{i,j} \oplus d_i), \tag{5.1}$$

jak bude ale dále uvedeno, nepodařilo se tajný klíč odhalit.

```
1 power_consumption = byte_Hamming_weight(after_sbox+1);
```

V pátém kroku je vytvořena smyčka, ve které je počítána hodnota vzájemné informace mezi naměřenými a hypotetickými proudovými průběhy.

```
1     parfor i=1:256
2         for j=1:n
3             key_trace(i,j)= ...
                  mutualinfo(traces(:,j),power_consumption(:,i));
4         end
```

Pro urychlení výpočtu je použita smyčka *parfor*, která v Matlabu spouští funkci pro paralelní výpočty. Celkový čas výpočtu je tak možné významně zkrátit.

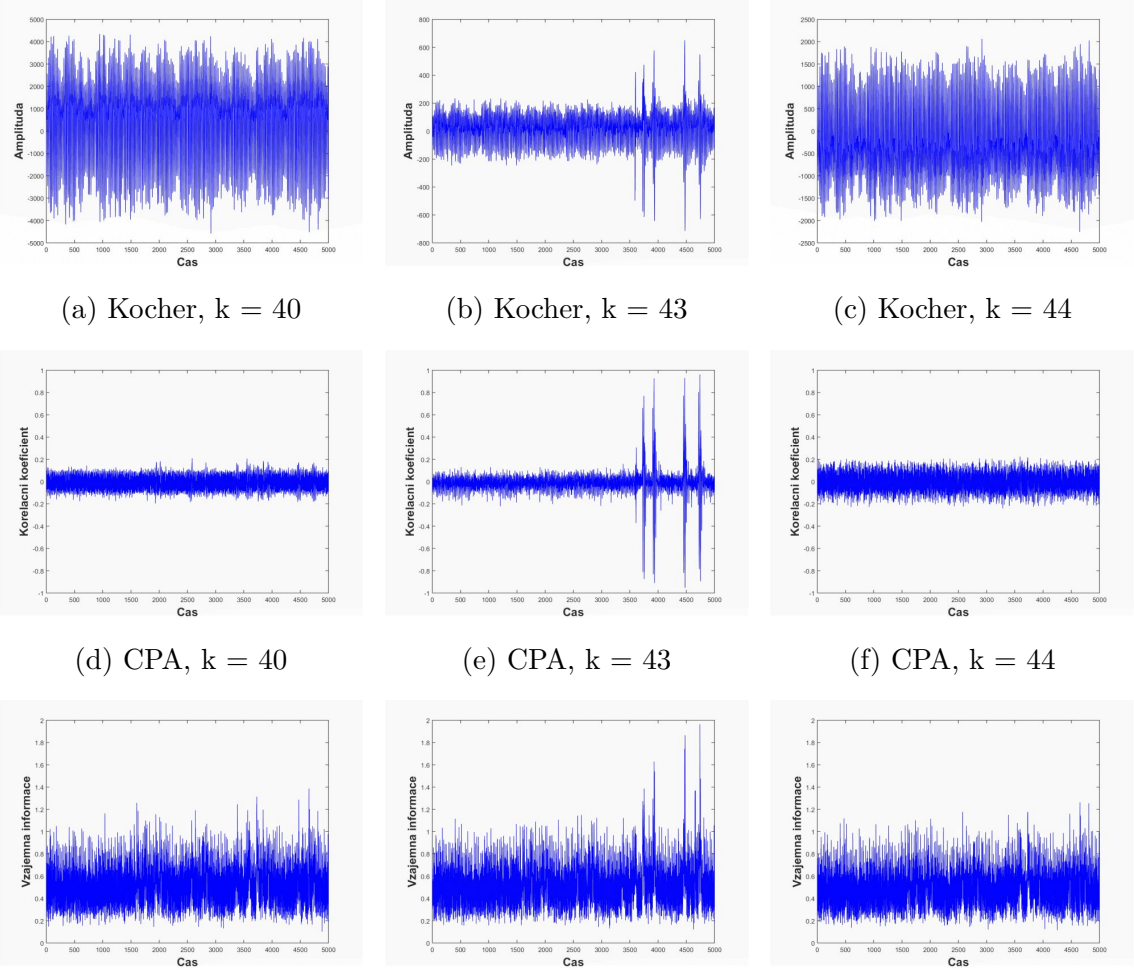
Závěrečná část slouží pro zobrazení grafu. Ve výstupních hodnotách je nalezena nejvyšší hodnota a podle jejich souřadnic vykreslí graf pro získanou hodnotu klíče.

```
1 [value, location] = max(key_trace(:));
2 [R,C] = ind2sub(size(key_trace),location);
3 figure
4 plot(key_trace(R,:));
5 title(['Graf hodnot vzajemne informace pro klic ',num2str(R)]);
6 disp(['Hodnota klice ',num2str(b),'. bytu je ',num2str(R),'.'])
```

5.1.2 Porovnání výsledků

Na Obrázku 5.1 jsou porovnány výsledky všech použitých analýz, Kocherovy metody, korelační proudové analýzy a analýzy vzájemnou informací. Jak je vidět útok pomocí analýzy vzájemné informace se zdařil. Na grafu pro hodnotu klíče 43 jsou výrazné vrcholy, ze kterých lze usoudit, že právě tato hodnota je tajným klíčem. To

je potvrzeno i ostatními dvěma metodami (u Kocherovy je to pouze hůř viditelné, díky dodržení měřítka). Z počtu vrcholů je možné odvodit, že klíč je používán ve více instrukcích.



(a) Kocher, k = 40 (b) Kocher, k = 43 (c) Kocher, k = 44

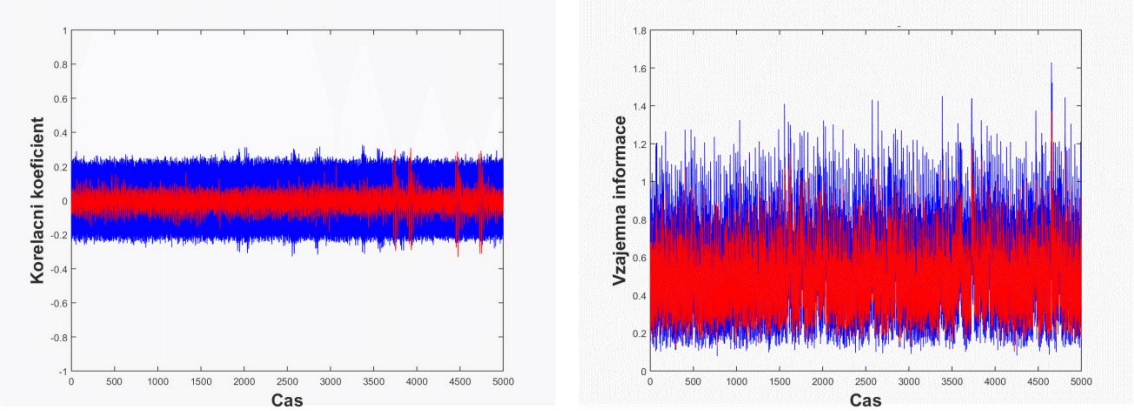
(d) CPA, k = 40 (e) CPA, k = 43 (f) CPA, k = 44

(g) MIA, k = 40 (h) MIA, k = 43 (i) MIA, k = 44

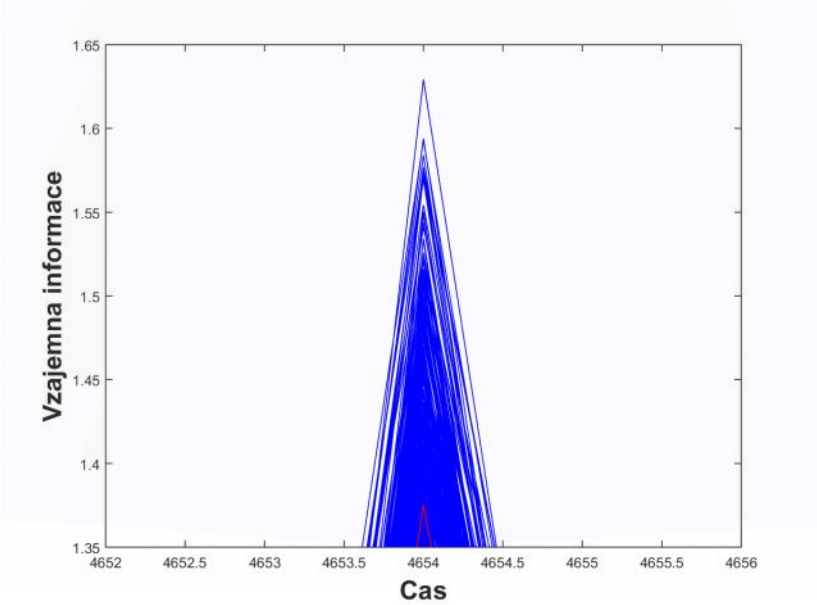
Obr. 5.1: Grafy vybraných klíčů Kocherovy metody, CPA a MIA

Na Obrázku 5.2 jsou výsledky pro klíč 43, který je skutečným tajným klíčem, hledaným pomocí CPA a MIA, ale pro proudový model Hammingovy vzdálenosti. Klíč se podařilo najít pouze pomocí korelační analýzy a to v čase $T = 4479$. V případě MIA byla maximální hodnota všech průběhů nalezena v čase $T = 4654$. Viz Obrázek 5.3. Jak je z něj patrné, tak hodnota správného tajného klíče je schovaná v mezi ostatními klíči.

Kompletní zdrojový kód skriptu je obsažen v příloze A.4 a na přiloženém DVD.



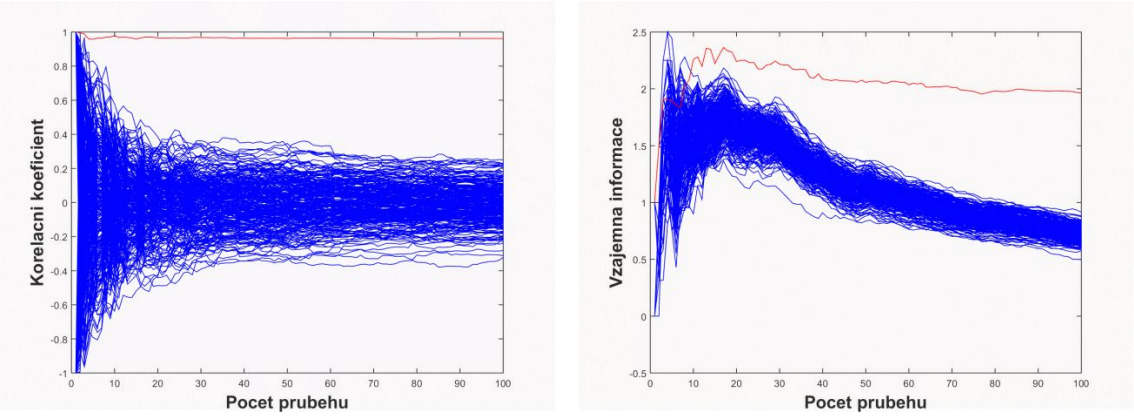
Obr. 5.2: Grafy klíčů průběhů hodnot klíčů pro CPA a MIA při použití Hammingovy vzdálenosti. Červeně je vykresleny hodnoty klíče 43.



Obr. 5.3: Graf hodnot MIA pro špičku v čase T = 4654. Graf hodnot klíče 43 je zobrazen červeně

5.1.3 Porovnání množství průběhů

Pro Korelační proudovou analýzu a Analýzu vzájemnou informací byl navržen skript, porovnávající množství průběhů potřebných k nalezení klíče. Tento skript je přiložen na DVD, jež je součástí této práce a zároveň je jeho plné znění v příloze A.5. Podle Obrázku 5.4 je možné srovnat výkon obou analýz. Ve skriptu byla konstanta *krok* nastavena na $krok = 2$, což znamená že počet průběhů bude ještě nutné vynásobit dvěma.



(a) (b)

Obr. 5.4: Grafy ukazující množství průběhů potřebných k nalezení tajného klíče. a) CPA, b) MIA

Jak je vidět pomocí analýzy CPA je možné zjistit tajný klíč již při použití desítky proudových průběhů. Pro MIA by bylo nutné použít alespoň dvojnásobku tedy 20 proudových průběhů, aby bylo dosaženo přesvědčivého výsledku.

5.1.4 Výsledky části DPAbook.org

V kapitole byly představeny metody DPA a s jejich pomocí se podařilo najít tajný klíč. Pro místo nálezu tajného klíče bylo vytvořeno srovnání metod CPA a MIA. Korelační proudová analýza v tomto případě potřebovala menší množství proudových průběhů nutných k odhalení klíče než Analýza vzájemnou informací.

Tyto skripty dobře slouží jako základ při studiu DPA a metod CPA a MIA.

5.2 DPA Contest 4.2

V této části diplomové práce je taktéž použita korelační proudová analýza a jako jeden z hlavních cílů práce je implementována analýza vzájemnou informací. Tentokrát však na průběhy dané v zadání této práce, tedy na průběhy ze soutěže DPA Contest 4.2. V úvodu je představeny významné části navržených skriptů. Dále je ukázáno řešení pro ideální vstupní podmínky, kdy jsou všechny proudové průběhy řádně synchronizovány a nejsou zatíženy šumem. V dalších částech je pak testováno postupné rozsynchronizování části průběhů (posun v časové ose) nebo navýšení šumu (posun v amplitudové ose). Tento postup je použit jak pro CPA, tak pro MIA. Poslední částí této kapitoly je pak další z hlavních cílů této diplomové práce a to porovnání výsledků výpočtů z pohledu množství proudových průběhů potřebných k odhalení tajného klíče.

5.2.1 Použité skripty a nastavení

V této části budou ukázány skripty navržené pro výpočty použitých statistických metod a prezentovány jejich výsledky.

Generátor chyb - `CPA_MIA_error_gen.m`

Z hlediska postupu řešení je nejvhodnější nejdříve představit generátor chyb. Je to z toho důvodu, že na jeho začátku jsou provedena nastavení všech konstant a počátečních hodnot potřebných proměnných. Další výpočty pak probíhají za stejných podmínek.

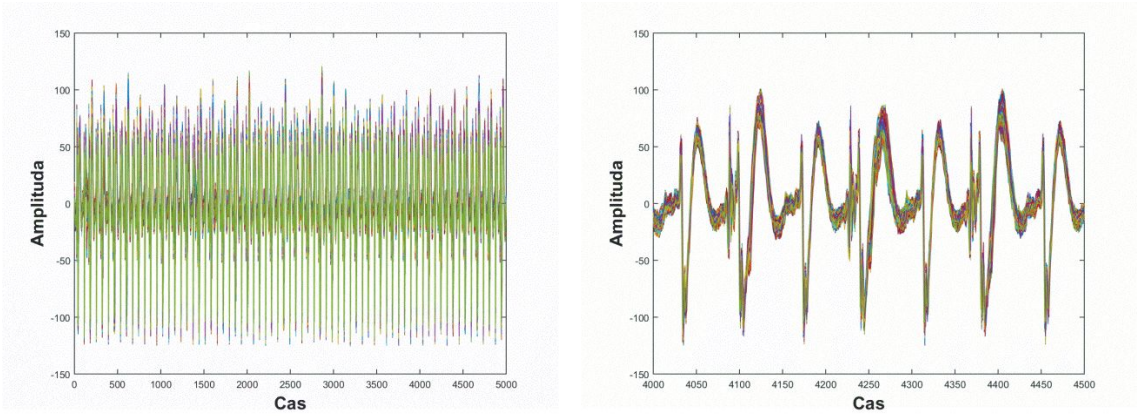
Po počátečním vyčištění prostředí Matlabu jsou zde první důležité proměnné a to *int_star*t a *int_start*.

```
1 int_start = 268000;
2 int_end = 273000;
```

S jejich pomocí je definován interval, na kterém budou probíhat výpočty. Pro hledání hodnoty klíče je vhodné nastavit větší rozsah vzhledem k počtu naměřených hodnot. Popřípadě vytvořit smyčku pro procházení celého rozsahu, který v našem případě byl větší než 1.700.000 hodnot. S prodloužením intervalu ale roste časová náročnost a nároky na operační paměť.

Tajný klíč byl nalezen v intervalu daném právě výše uvedenými dvěma hodnotami. Všechny následující grafy jsou pak zobrazeny s hodnotami danými tímto rozsahem, nebo v případě potřeby detailnějšího zobrazení s rozsahem nižším.

Na Obrázku 5.5a jsou zobrazeny vstupní proudové průběhy z rozsahu uvedeného v této kapitole a detail části, z které budou vycházet další zobrazení v této kapitole.



(a) (b)

Obr. 5.5: Graf vstupních proudových průběhů. a) interval 268000-273000, b) interval 272000-272500

V další části je nastaveno množství proudových průběhů určených k výpočtům a jejich načtení. Počet zpracovávaných průběhů je dán počtem použitých souborů z DPA Contest 4.2, přičemž každý z nich obsahuje data o 1000 průbězích. V našem případě jsou tyto soubory použity dva, bude tedy zpracováváno 2000 proudových průběhů. Zpracovaná vstupní data jsou pak uložena do souboru `DPA_var.mat`.

Na počátku další části jsou příkazy pro generování šumu. Průběh je zašuměn pomocí funkce pro aditivní bílý gaussovský šum *awgn*. Je u něj měněna hodnota SNR, tedy odstupu signálu od šumu. Vstupní data jsou nejprve přetypována na typ *double*, protože hodnoty generované funkcí *awgn* jsou také typu *double*. Jejich hodnoty je potom nutné zaokrouhlit na celá čísla, jelikož pouze ta jsou zpracovatelná při výpočtu vzájemné informace. To je provedeno pomocí funkce fix, která zaokrouhluje čísla směrem k nule. Výsledné průběhy budou zajímavější než kdyby byly jen prostě zaokrouhleny. Následně je typ změněn na původní $int8$.

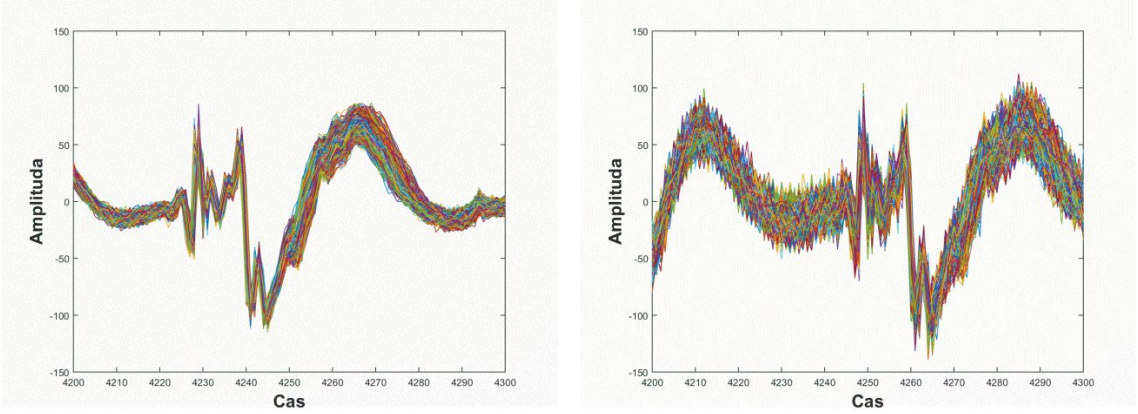
```
for SNR = 40:-2:2

    ...

    traces_all = double(traces_all);
    for sumR = 1:num_traces_testing
        traces_all(sumR,:) = ...
            awgn(traces_all(sumR,:),SNR,'measured','db');
    end

    traces_all = fix(traces_all);
```

```
11      traces_all = int8(traces_all);
12
13      ...
14
15  end
```



(a) (b)

Obr. 5.6: Grafy vstupních proudových průběhů. a) základní bez šumu, b) SNR = 14 dB

Skript dále nastavuje hodnotu posunutí počátečního bodu pro určité procento vstupních proudových průběhů. Konstanta *rozptyl* určuje maximální absolutní hodnotu posunu řádků vstupních hodnot. Čísla řádků jsou pak náhodně seřazena (*err_lines*). Proměnná *err_vec* obsahuje čísla od nuly do sta s krokem pět, pomocí kterých je definováno množství posunutých průběhů. Náhodné posunutí řádků je zajištěno proměnnou *posun*. Ta nabývá hodnot <-rozptyl, rozptyl>, tedy může být i nulová. Množství skutečně posunutých proudových průběhů je potom o něco menší než hodnota daná hodnotou z *err_vec*. Na výsledek to má ale minimální vliv.

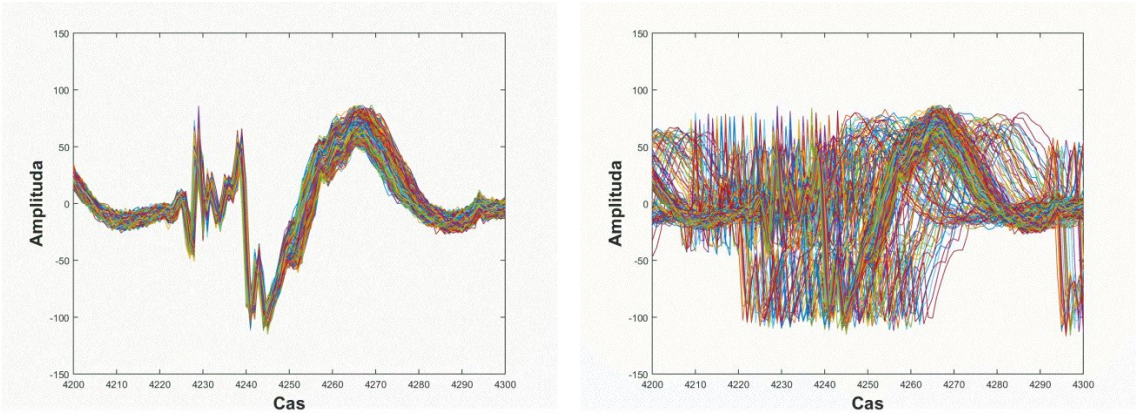
```
rozptyl = 20;
...
err_lines = randperm(num_traces_testing);
err_vec = 0:5:100;
posun = randi([-rozptyl rozptyl],1,num_traces_testing);
...
```

Všechny takto vygenerované hodnoty jsou využity v následující smyčce.

```
    for hlavni_cyklus = 1:21
        ...
```

```
3           asyn_lines = err_vec(hlavni_cyklus)*(num_traces_testing/100);
4           do_err = err_lines(1,1:asyn_lines);
5           do_err = sort(do_err);
6
7           traces_S = [];
8           traces_S = traces_all(:,rozptyl+1:rozdil+rozptyl+1);
9
10          for i = 1:asyn_lines
11              err_line_1st = traces_all(do_err(1,i),:);
12              err_line_2nd = err_line_1st(1,1 + rozptyl + ...
                    posun(1,i):rozdil + 1 + rozptyl + posun(1,i));
13              traces_S(do_err(1,i),:) = err_line_2nd;
14          end
15      ...
16      end
```

Nejdříve je určeno množství posunutých řádků a vzestupně seřazeny jejich indexy. Interval vstupních proudových průběhů je rozšířen na obě strany o hodnotu rozptylu a z těchto průběhů jsou potom podle indexu vybrány zadané řádky, s počátkem určeným hodnotou rozptylu. Výsledná vstupní data pak vypadají jako například ta na Obrázku 5.7.



(a) (b)

Obr. 5.7: Grafy vstupních proudových průběhů. a) základní bez posunu, b) posunuto 5 % průběhů

Pro nastavení průběhů bez chyb je potřeba nastavit proměnné $SNR = 100$ a $hlavni\_cyklus = 1$. Tím je šum nastaven s odstupem 100 dB od signálu a zároveň jsou všechny průběhy synchronizovány, jelikož první hodnota v proměnné *hlavni_cyklus* je nula.

Nyní jsou připravena všechna data nutná k hledání klíče, které je zpracováno v souboru `CPA_MIA_uknownKey.m`.

Skript hledání klíče

Hledání klíče je cílem skriptu `CPA_MIA_uknownKey.m`. Na začátku skriptu jsou dva cykly, každý pro jednu z metod. Díky tomu, že jsou všechna důležitá nastavení provedena již v předchozím kroku, tak výpočet probíhá se stejnými vstupní daty pro obě metody.

```
for metoda = 1:2
    if metoda == 1
        method = ('correlation');
        disp('Porovnavam - CA');
    else
        method = ('mutual');
        disp('Porovnavam - MIA');
    end
    ...
end
```

Soubor `Tabulky_n.mat` obsahuje doplňující proměnné, jako například model Hammingovy váhy nebo výsledky hledání tajného klíče. Proměnná *krok* vyjadřuje periodickou hodnotu pořadí prvku použitého při výpočtu. Pro všechny grafy v této práci zobrazené je nastavena hodnota $krok = 10$. Nastavení této proměnné ovlivňuje přesnost výpočtů. Platí, že čím delší *krok* bude, tím budou výpočty méně přesné, ale budou probíhat rychleji.

V další části probíhá výpočet hodnot podle algoritmu AES-128. Podrobně je popsán například v [6], [2], případně zde[2].

Následuje část výpočtu statistických metod. Tato část odpovídá pátému bodu řešení DPA, jak je uvedeno v kapitole 2.2.

```
for i = 1:256
    switch method
        case 'correlation'
            corr_mat = corrcoef([pre  hw_xor_sbox_in_out(:,i)]);
            [m,n] = size(corr_mat);
            dpa_result(i,:) = corr_mat(1:n-1,m);
        case 'mutual'
            [m,n] = size(pre);
            fprintf(1,'key %d\n',i);
            for j=1:n
                dpa_result(i,j)= ...
                    mutualinfo(pre(:,j),hw_xor_sbox_in_out(:,i));
            end
```

[2]https://en.wikipedia.org/wiki/Advanced_Encryption_Standard

```
13      end
14  end
```

Ze zápisu je vidět, že výpočet probíhá pro všechny klíče 0 až 255[3]. V první části se řeší výpočet pro korelační proudovou analýzu. Je použita funkce *corrcoef*, která je součástí balíku funkcí v Matlabu. Funkce *corrcoef* vypočítá korelační koeficient mezi dvěma proměnnými, v tomto případě mezi hypotetickými hodnotami proudové spotřeby a hodnotami vstupních proudových průběhů. Z výpočtů je nakonec sestavena matice korelačních koeficientů, z kterých je na základě nalezení hodnoty nejvíce vzdálené od nuly určen hledaný tajný klíč.

V druhé části je zadání pro výpočet analýzy vzájemnou informací. Syntaxe je podobná té pro korelační analýzu. Stejně jako v kapitole 5.1 je i zde použita funkce `mutualinfo.m`. Tato funkce je aplikací výpočtu z kapitoly A.6, konkrétně je použita zvýrazněná část vzorce 2.7.

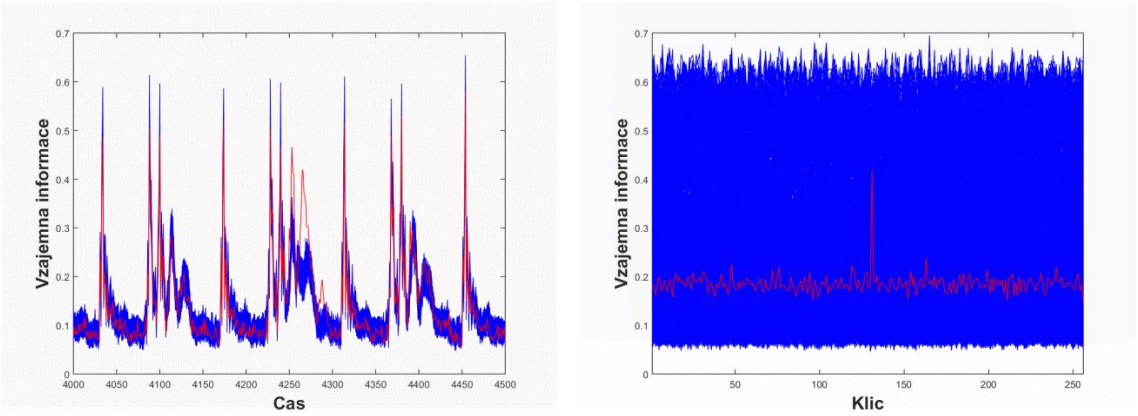
$$I(X;Y) = H(X) + H(Y) - H(X,Y) \tag{5.2}$$

Úprava pro MIA

V případě výpočtu MIA není na první pohled možné určit tajný klíč nalezením nejvyššího vrcholu, viz Obrázek 5.8a. Na Obrázku 5.8b je srovnání z pohledu klíčů. Jak je vidět, klíč je "utopen"v záplavě ostatních hodnot. Pokud ale zvětšíme graf v některé z nejvyšších špiček, zjistíme, že jsou v těchto místech hodnoty velmi blízko u sebe, Obrázek 5.9 (Falešný vrchol), oproti místu kde jeden z průběhů významně dominuje (Tajný klíč). Pro úpravu výsledků můžeme použít buď střední hodnotu nebo medián. V této práci je použit medián. Ten je vypočítán pro každý časový okamžik a odečten od výsledných hodnot každého z průběhů v daném okamžiku. Falešné vrcholy zmizí a zůstanou pouze špičky pro tajný klíč, viz Obrázek 5.10b. To, že jsou hodnoty výrazně poníženy není v tuto chvíli nijak podstatné. Zásadní v tuto chvíli je, že tato úprava dovolila najít vrcholy, ze kterých je možné určit hodnotu klíče. To je možné např. z Obrázku 5.10d. Po provedení korekce vyjde hodnota tajného klíče 130.

Kód pro odpočet mediánu je níže. Úprava je použita pouze pro nalezení klíče. V části, kdy budeme porovnávat množství potřebných průběhů se tato úprava nepoužívá, protože není nutná, jelikož výpočty probíhají pouze v konkrétním bodě průběhu.

[3]Skutečný rozsah všech klíčů je 0 až 255 ($2^8 - 1$) klíčů. Číslování v Matlabu vychází z čísel sloupců a řádků, které vždy začínají od jedničky. Odpočet je tedy nutné udělat dodatečně.



(a) (b)

Obr. 5.8: MIA - Grafy výstupu výpočtu pro určení klíče. a) zobrazení z pohledu času, b) zobrazení z pohledu klíčů. Červenou barvou je vyznačen průběh tajného klíče

```
medCPA=median(CPA);
[y,x] = size(CPA);
for i = 1:x
    graf_medCPA(:,i)=CPA(:,i)-medCPA(i);
end
```

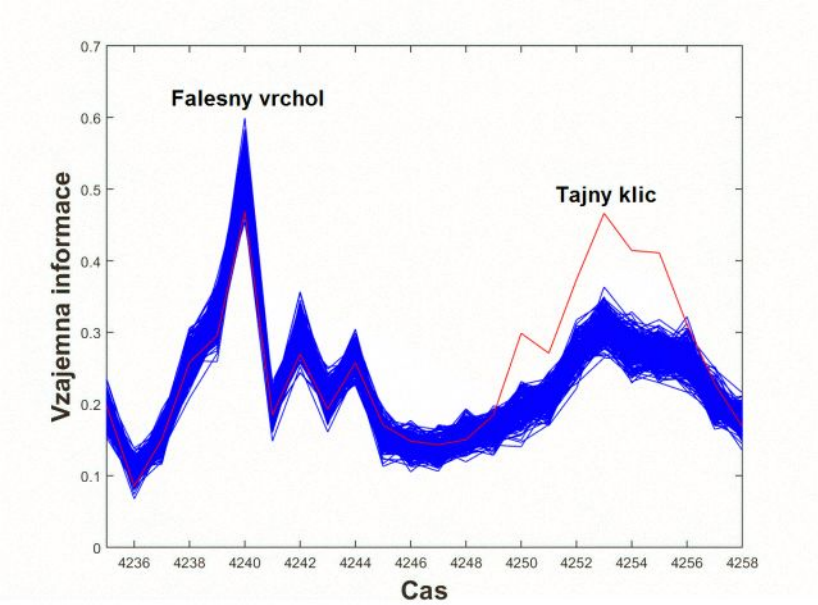
Skript porovnání množství proudových průběhů nutných k odhalení klíče

Posledním skriptem používaným v práci je `CPA_MIA_compareCountTraces.m`. Tento skript vychází z předchozího. Výstupem z něj je graf, z kterého lze určit kolik proudových průběhů je potřeba ke zjištění tajného klíče. Skript pracuje tak, že v každé iteraci načte počet proudových průběhů daný proměnou *krok* a zapíše výsledky pro každou hodnotu klíče. Se vzrůstajícím množstvím proudových průběhů se začne průběh tajného klíče osamostatňovat. I zde můžeme proměnnou *krok* ovlivňovat přesnost a rychlost výpočtu. Opět zde platí výše popsané.

```
for iteration = krok:krok:num_traces_testing
...
end
```

5.2.2 Analýza zadaných průběhů

Veškerá výstupní data jsou přehledně shrnuta na Obrázku 5.10. První řádek (5.10a a 5.10b) ukazuje detail výstupních průběhů, místo kde se nachází maximální hodnota



Obr. 5.9: Neupravený výstup MIA - porovnání špičkových průběhů

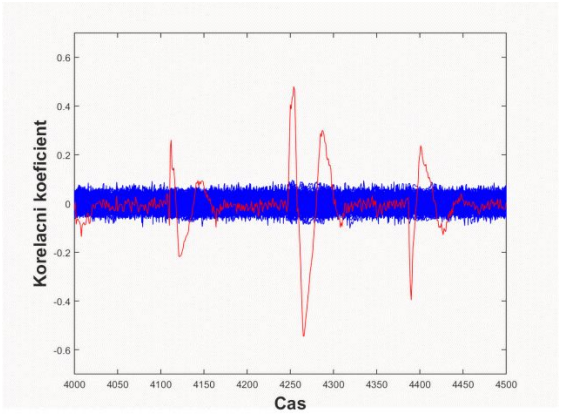
pro CPA a MIA. Pro CPA může jít vlastně i o hodnotu minimální, jak plyne z teorie korelačního koeficientu. Pro obě funkce jde téměř o stejný bod v čase. U CPA je to čas $T = 4266$, pro MIA jde o čas $T = 4265$. V těchto časech budou počítány veškeré další analýzy pro proudové průběhy zatížené chybami.

Pokud transformujeme matici výsledků maximální špičky, zjistíme hodnotu tajného klíče, která je zobrazena v druhém řádku (Obrázek 5.10c, 5.10d). Jak již bylo uvedeno na straně 34, hodnota klíče je 130.

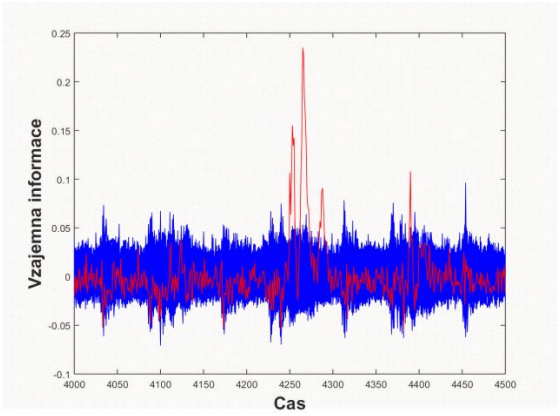
Na Obrázku 5.11 je potom možné srovnat úspěšnost použitých metod. Výpočty byly provedeny pro 2000 vstupních proudových průběhů, při výpočtu byl použit *krok* = 10. Grafy na Obrázcích 5.11a a 5.11b proto mají na ose x maximální hodnotu 200. Jak je vidět, pro situace kdy jsou použita vstupní data bez jakéhokoliv zkreslení, je úspěšnější analýza pomocí korelačního koeficientu. Z grafu je jasně viditelné, že průběh hodnot tajného klíče je bez problémů odlišitelný od ostatních již před bodem $T = 20$, což znamená, že pro odhalení tajného klíče pomocí CPA stačí méně než 200 proudových průběhů. U analýzy pomocí vzájemné informace se klíč začíná odlišovat až výrazně za $T = 20$, odhadem okolo $T = 30$. Z toho jasně plyne, že pro MIA je nutné v tomto případě zajistit 300 a více průběhů, aby byl klíč nalezen.

5.2.3 Analýza průběhů ovlivněných chybami

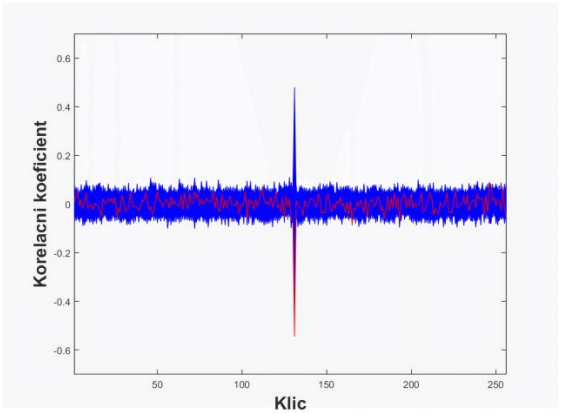
Dostáváme se do části, kde bude ukázáno jak si obě analytické metody poradí s chybami ovlivněnými vstupními proudovými průběhy. Těmito chybami bude šum a špatná synchronizace. Šum bude nastavován změnou parametru SNR (Signal to



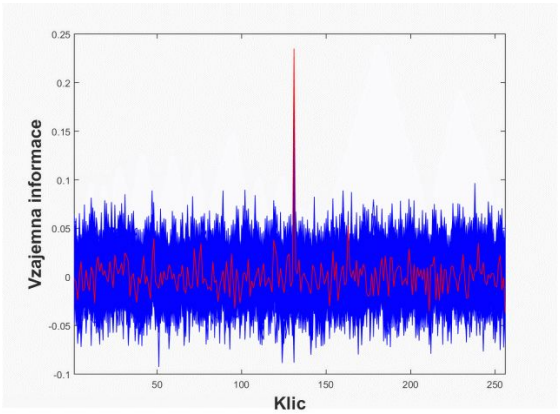
(a) CPA - maximální špička



(b) MIA - maximální špička



(c) CPA - hodnota klíče



(d) MIA - hodnota klíče

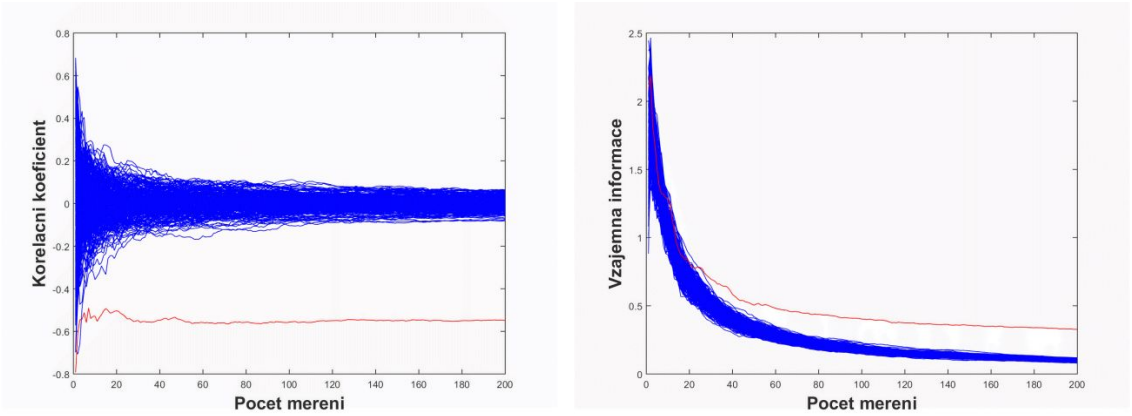
Obr. 5.10: Grafy pro vypočtené hodnoty výstupů bezchybových průběhů

Noise Ratio), špatné synchronizace bude dosaženo posouváním počátku času měření určitého procenta vstupních proudových průběhů vzhledem k jejich celkovému množství. Jak již bylo dříve uvedeno, skripty je zpracováváno 2000 proudových průběhů. Rozsynchronizováním 10 % dojde tedy k posunu u dvou set z nich.

Předpokládaným výsledkem dopadu chyb je snížení maximální výšky špičky průběhu hodnot tajného klíče. Ta by se měla se vzrůstajícím chybovostí snižovat a od nějaké úrovně zmizet mezi ostatními průběhy. Tím by mělo být znemožněno nalezení tajného klíče.

Vliv šumem ovlivněných proudových průběhů na výsledky statistických analýz

Generování šumu probíhá ve skriptu `CPA_MIA_error_gen.m`. Změna velikosti šumu je nastavována pomocí změny velikosti proměnné SNR. Hodnoty SNR byly nastavovány v rozmezí od 40 dB do 8 dB s krokem -8 dB. Toto zobrazení postačuje k zob-



(a) CPA - množství proudových průběhů (b) MIA - množství proudových průběhů

Obr. 5.11: Vizualizace počtu nutných proudových průběhů pro metody CPA a MIA

razení změn v množství potřebných vstupních proudových průběhů, jelikož se toto se zvyšujícími se hodnotami SNR už nemění. Hodnota proměnné *hlavni_cyklus* zůstane rovna jedné. Výsledky některých těchto výpočtů jsou na Obrázku 5.13 a 5.14.

Nejvyšší absolutní rozdíl hodnot je u CPA v nejnižším bodě průběhu tajného klíče. Pro porovnání bude použit tedy tento bod. Na levém sloupci Obrázku 5.13 je vidět, že absolutní hodnota špičky průběhu hodnot tajného klíče je skutečně šumem ovlivněna podle předpokladu a s klesající hodnotou SNR také klesá. Počet proudových průběhů, které budou nutné k nalezení tajného klíče tak stoupá. Z počátečních desítek[4] průběhů u $SNR = 40\ dB$, po stovky u $SNR = 8\ dB$.

U MIA se vychází z nejvyšší špičky. V levém sloupci Obrázku 5.14 je také vidět, že maximální hodnot špičky s klesající hodnotou SNR také klesá. V pravém sloupci je také vidět jak křivka hodnot průběhu tajného klíče pomalu klesá směrem k ostatním, až mezi nimi nakonec zmizí.

Nejdůležitější zjištění této části je, že **CPA je odolnější proti šumu než MIA**.

Vliv špatné synchronizace proudových průběhů na výsledky statistických analýz

Pro to aby DPA podávala správné informace, je nutné dodržovat správnou časovou synchronizaci všech proudových průběhů. Pokud není dodržena, může vést stejně jako šum k nemožnosti nalezení tajného klíče.

Rozsynchronizování proudových průběhů by mělo stejně jako šum vést ke snížení výšky maximální špičky. Je vytvořeno jejich posunutím ve směru časové osy vpravo

[4]Jak bylo popsáno v kapitole o skriptu `CPA_MIA_compareCountTraces.m`, $krok = 10$

nebo vlevo. K tomu dochází u části průběhů dané proměnnou *err_vec* a to od 0% do 65%. Například při 60 % průběhů bude nějak posunuto až 1200 řádků obsahujících vstupní data. Maximální posun je dán hodnotou konstanty *posun*, která vychází z přibližné šířky vrcholu vstupního proudového průběhu v bodě, kde se nachází maximální hodnota tajného klíče. Viz Obrázek 5.7a.

Výsledky výpočtů CPA a MIA jsou na Obrázcích 5.15 a 5.16. I tady je v jejich levých sloupcích vidět, jak se špička zmenšuje a přibližuje směrem k ostatním průběhům. Výsledek pravé části je však ve výsledku odlišný od toho jak vypadaly grafy ovlivněné šumem. I u posledního grafu, kde je rozsynchronizováno 60 % tedy 1200 vstupních proudových průběhů, lze pomocí MIA nalézt tajný klíč přibližně po použití 1100 průběhů. U CPA je tomu již při daleko menším počtu. Už u 30 % rozsynchronizovaných průběhů je obtížné klíč nalézt. Daří se to téměř až u maximálního počtu, někde okolo 1900 proudových průběhů. Z toho je možné vyvodit závěr, že **MIA je odolnější proti rozsynchronizování než CPA**.

5.2.4 Porovnání analytických metod

Jak bylo v předchozích kapitolách uvedeno, každá z použitých metod je odolnější na jiný druh chyb. CPA je odolnější proti šumu, MIA lépe zvládá chyby v synchronizaci. Na Obrázku 5.12a jsou zobrazeny křivky srovnávající odolnost porovnávaných metod proti šumu. Je zde graficky znázorněn výsledek kapitoly zabývající se šumem. Hodnoty křivek v grafu byly odhadnuty na základě grafů porovnávajících množství vstupních proudových průběhů. Jejich část je zobrazena na Obrázcích 5.13 a 5.14. Odhady jsou zapsány v Tabulce 5.1.

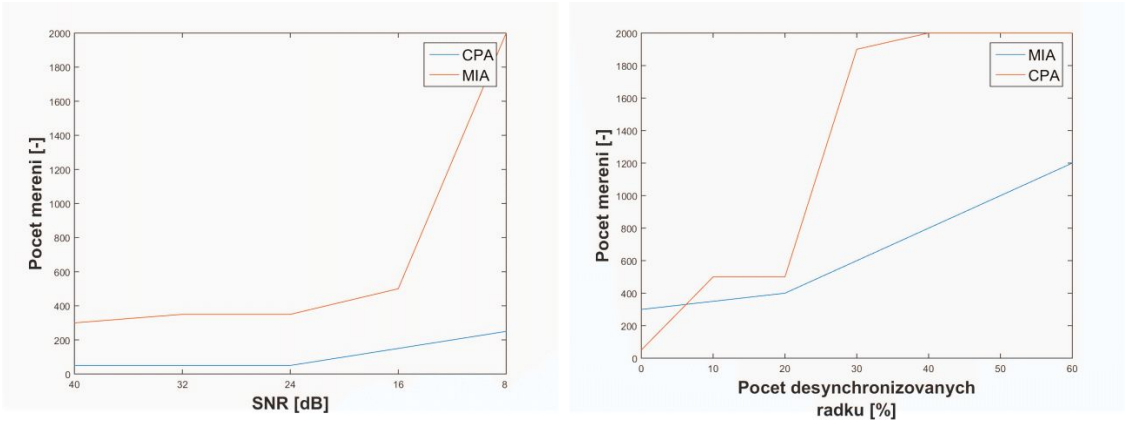
SNR	dB	40	32	24	16	8
CPA	-	50	50	50	150	250
MIA	-	300	350	350	500	2000

Tab. 5.1: Tabulka počtu vstupních proudových průběhů pro měnící se SNR

Desync	%	0	10	20	30	40	50	60
CPA	-	50	500	500	1900	2000	2000	2000
MIA	-	300		400		800		1200

Tab. 5.2: Tabulka počtu vstupních proudových průběhů pro měnící se procento desynchronizovaných průběhů

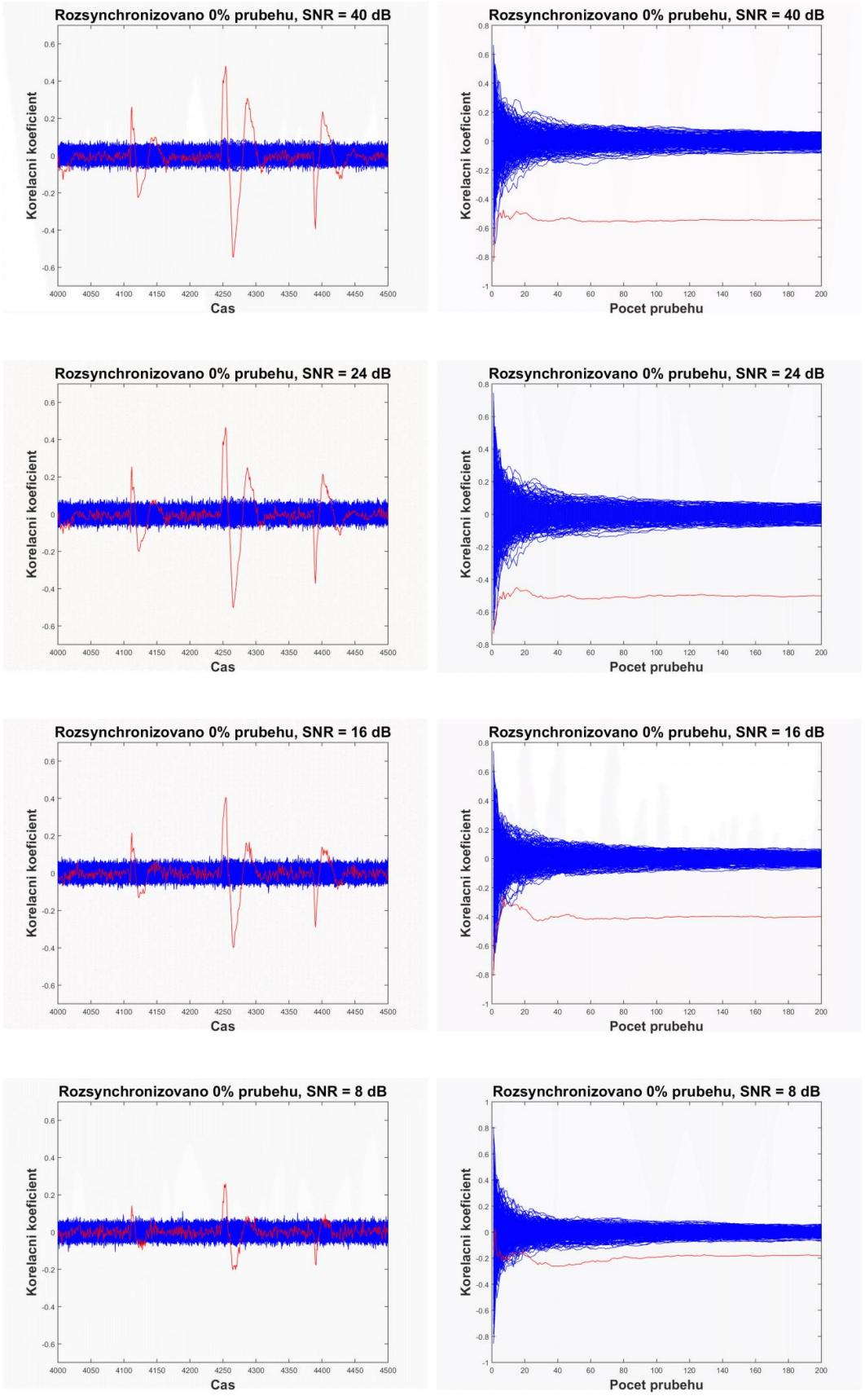
Jak je zde vidět, Korelační proudová analýza je v případě proudových průběhů zatížených šumem vždy úspěšnější při hledání tajného klíče než Analýza vzájemnou informací.



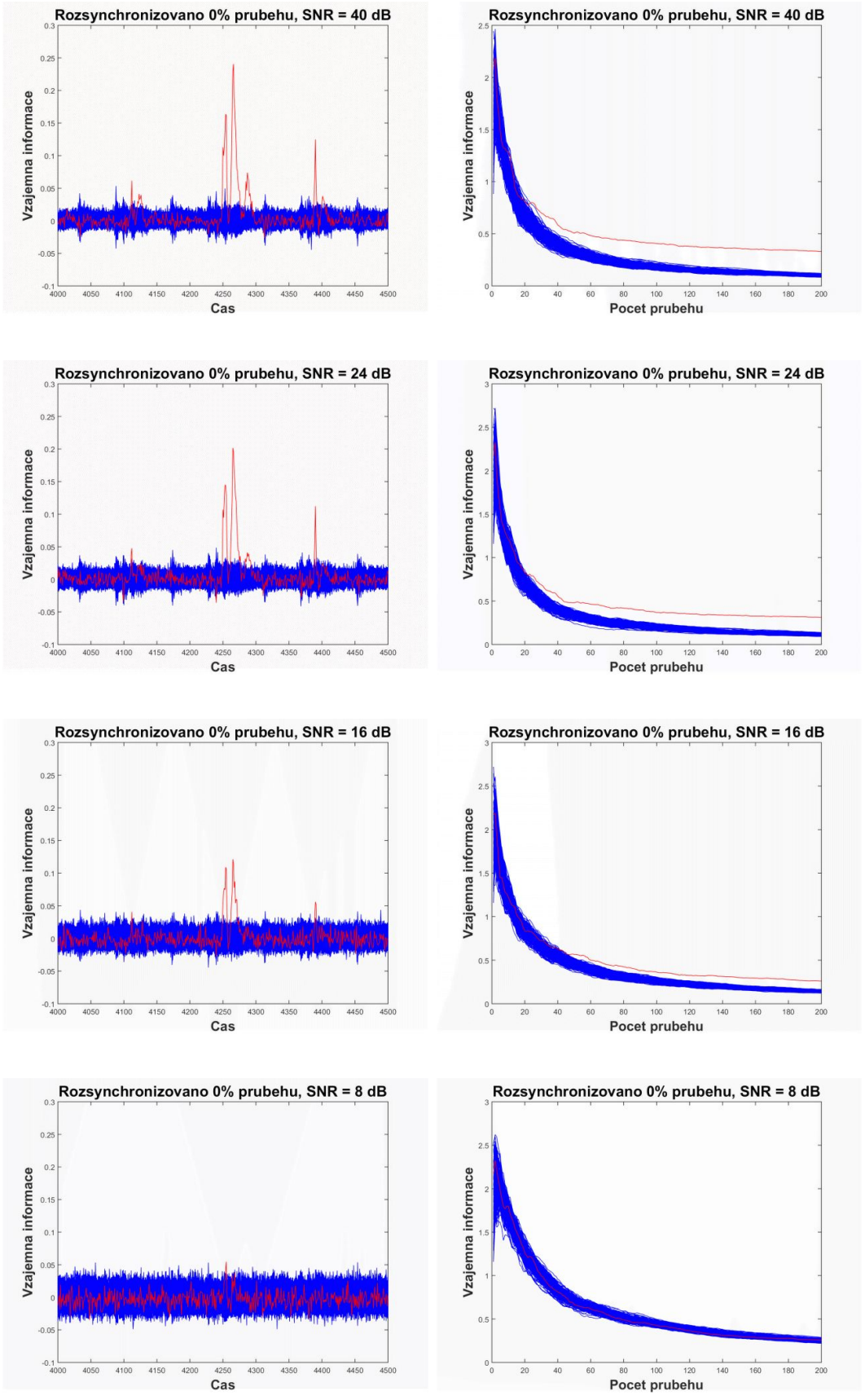
(a) (b)

Obr. 5.12: Grafy ovlivnění výsledků CPA a MIA chybami. a) Šum, b) Desynchronizace

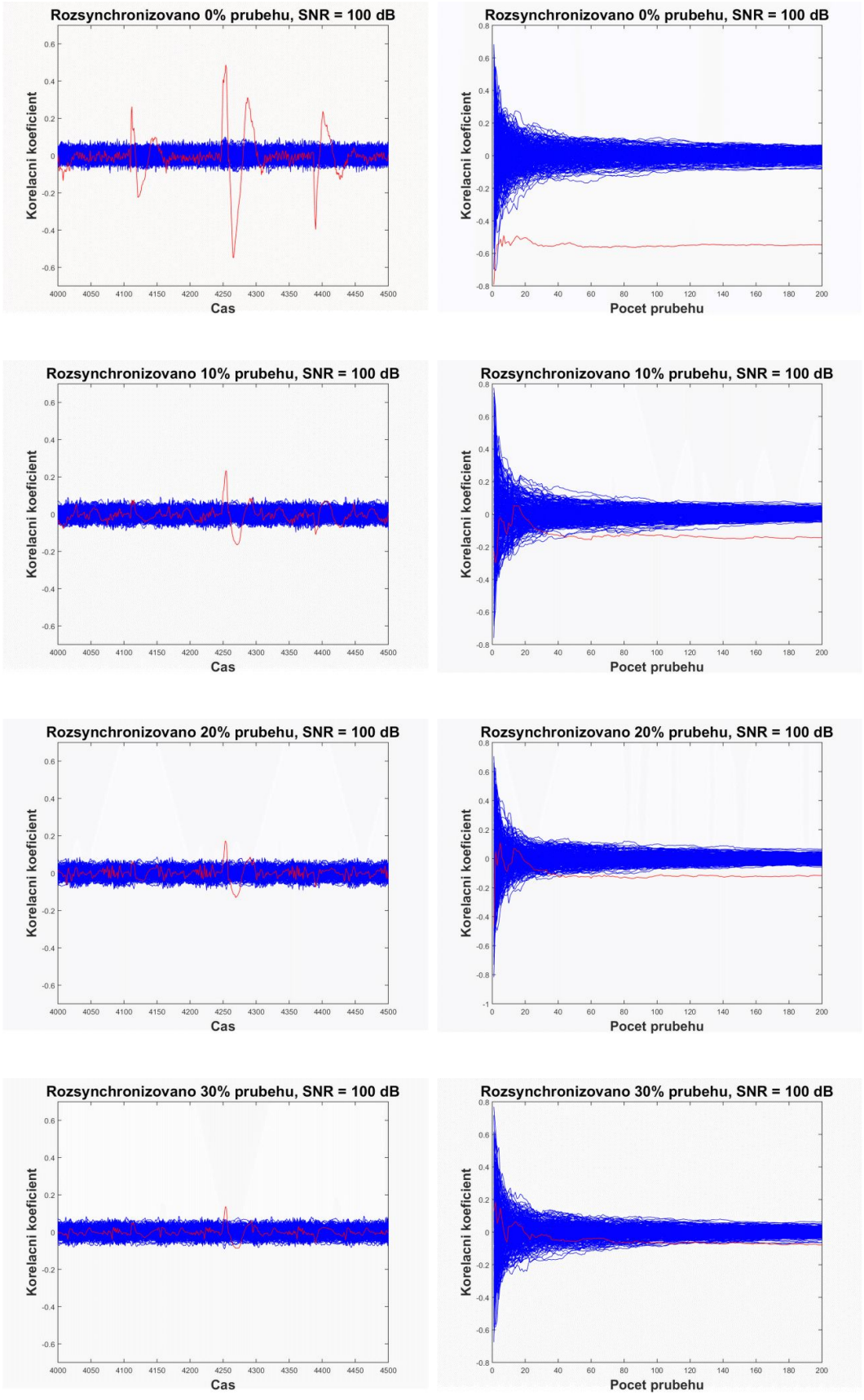
V případě desynchronizovaných průběhů je tomu od určitého množství průběhů naopak. Korelační proudová analýza potřebuje k nalezení tajného klíče méně vstupních proudových průběhů přibližně do šesti procent desynchronizovaných vstupních průběhů. Od tohoto bodu nároky prudce stoupají. Analýza vzájemnou informací od tohoto bodu potřebuje k nalezení klíče méně vstupních proudových průběhů. Jejich grafické srovnání je na Obrázku 5.12b. Data pro tento graf jsou v Tabulce 5.2.



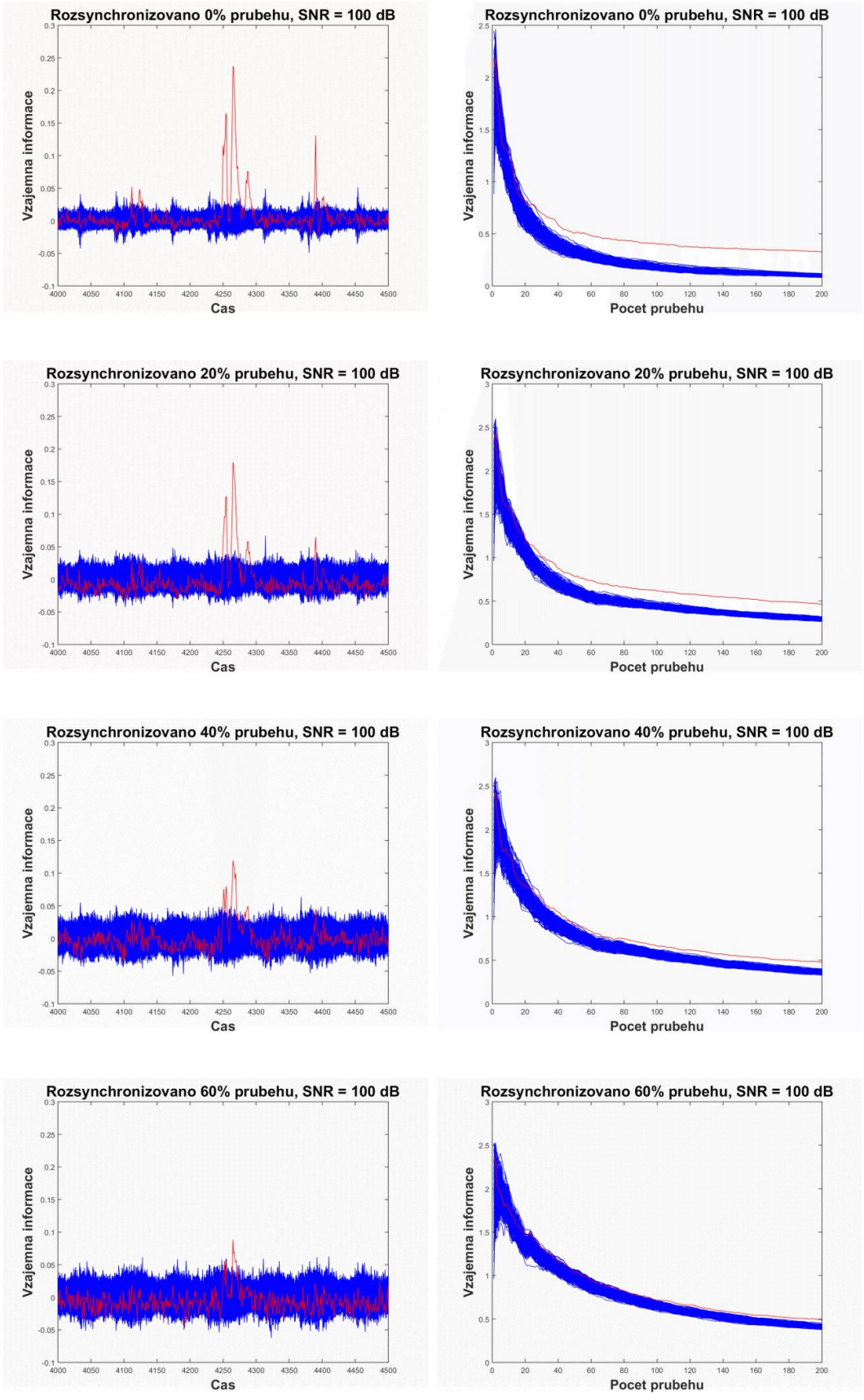
Obr. 5.13: Grafy ovlivnění výsledků CPA šumem



Obr. 5.14: Grafy ovlivnění výsledků MIA šumem



Obr. 5.15: Grafy ovlivnění výsledků CPA posunem



Obr. 5.16: Grafy ovlivnění výsledků MIA posunem

6 ZÁVĚR

V rámci diplomové práce byla studována problematika útoků pomocí neprofilujících útoků diferenciální proudovou analýzou. V práci jsou uvedeny některé typy útoků proudovou analýzou a to korelační proudová analýza a analýza vzájemnou informací. Hlavní náplní práce byla implementace útoku analýzou vzájemné informace v prostředí Matlab na průběhy soutěže DPA Contest 4.2 a porovnání výsledků metody z pohledu efektivity útoku. Tato efektivita měla být vyjádřena množstvím proudových průběhů, potřebných k úspěšně provedenému útoku.

V úvodu práce jsou zmíněny některé techniky sloužící k útoku pomocí DPA, některé možnosti obrany proti nim a stručný popis algoritmu AES. V další části je nejdříve představen jednoduchý skript sloužící k útokům pomocí DPA (původně součást *dpabook.org*). Do tohoto skriptu byla implementována Analýza vzájemnou informací. Byl navržen skript umožňující srovnání množství průběhů. Tyto skripty posloužily jako vědomostní základ pro hlavní cíle této diplomové práce.

Pro splnění hlavních cílů byl upraven skript, které umožňují najít a určit hodnotu tajného klíče pro vstupních proudové průběhy dle DPA Contest 4.2 a upraven skript pro zjištění množství potřebných vstupních proudových průběhů. Dále byl navržen skript, umožňující generování chyb do vstupních proudových průběhů a to jak ve směru časové osy tím, že posouvá počátek průběhu v daném rozsahu, tak mění náhodně amplitudu generováním šumu do vstupních proudových průběhů. Tím tak bylo možné různě nastavovat veškerá vstupní data nutná pro analýzu množství průběhů. V práci jsou rozebrány hlavní části použitých skriptů.

U metod CPA a MIA byla testována jejich schopnost nalézt tajný klíč při působení šumu a desynchronizace na vstupní proudových průběhů. Výsledkem těchto testů je zjištění, že Korelační proudová analýza dosahuje správných výsledků i u vstupních proudových průběhů ovlivněných šumem. Její nevýhodou může být, že je málo odolná proti špatné synchronizaci. Pokud je použita na nedostatečně zesynchronizované vstupní proudové průběhy, bude jich potřebovat tím více, čím více jich bude rozsynchronizováno.

Naproti tomu Analýza vzájemnou informací vyžaduje velké množství vstupních proudových průběhů v případě, že budou ovlivněny šumem. Více než CPA. V případě desynchronizovaných proudových průběhů dosahuje ale překvapivě mnohem lepších výsledků než CPA.

Každá z použitých analýz je vhodná pro jiné případy. Výsledky z obou těchto částí jsou shrnuty na Obrázku 5.12.

Všechny cíle dané zadáním diplomové práce se podařilo splnit. Práci je možné dále rozvíjet přidáváním dalších statistických metod, přičemž jejich implementace měla být jednoduchá při použití skriptů, jež tato práce obsahuje.

LITERATURA

[1] KOCHER P., JAFFE J., JUN B. *Differential Power Analysis. In CRYPTO '99: Proceedings of the 19th Annual International Cryptology Conference on Advances in Cryptology.* London, UK: Springer-Verlag, 1999, s. 388–397, ISBN 3-540-66347-9. Dostupné z URL: <http://link.springer.com/chapter/10.1007%2F3-540-48405-1_25>.

[2] MANGARD, S., OSWALD E., POPP T. *Power Analysis Attacks: Revealing the Secrets of Smart Cards (Advances in Information Security).*, Secaucus, NJ, USA:Springer-Verlag New York, Inc., 2007, ISBN 0-387-30857-1.

[3] SAKIYAMA K., SASAKI Y., LI Y. *Security of Block Ciphers: From Algorithm Design to Hardware Implementation.*, 1st Edition, John Wiley and Sons Inc, 2015 ISBN 1-118-66001-3

[4] DAEMEN J., RIJMEN V. *The Design of Rijndael: AES - The Advanced Encryption Standard.*, Information Security and Cryptography. Springer, 2002, ISBN 3-540-42580-2.

[5] PENG H., LONG F., DING C., *Feature selection based on mutual information: criteria of max-dependency, max-relevance, and min-redundancy.*, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 27, No. 8, pp.1226-1238, 2005.

[6] MARTINÁSEK Z. *Kryptoanalýza postranními kanály.* [Disertační práce.] Brno: VUT, FEKT, 2013. 129 s.

[7] HNAT W., PETTENGILL J. *Differential Power Analysis Side-Channel Attacks in Cryptography.* [A Major Qualifying Project.] Worcester Polytechnic Institute, 2010. 41 s.

[8] KOCHER P. *Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In CRYPTO '96 Proceedings of the 16th Annual International Cryptology Conference on Advances in Cryptology.*, London, UK: Springer-Verlag, 1996, s. 104-113, ISBN 3-540-61512-1. Dostupné z URL: <http://courses.csail.mit.edu/6.857/2006/handouts/TimingAttacks.pdf>.

[9] QUISQUATER J.-J., SAMYDE D. *Electromagnetic analysis (ema): Measures and counter-measures for smart cards, Proceeding E-SMART '01 Proceedings*

of the International Conference on Research in Smart Cards: Smart Card Programming and Security, London, UK: Springer-Verlag, 2001, s. 200–210, ISBN 3-540-42610-8

[10] GANDOLFI K., MOURTEL C., OLIVIER F. *Electromagnetic analysis: Concrete results, in CHES '01: Proceedings of the Third International Workshop on Cryptographic Hardware and Embedded Systems.* London, UK: Springer-Verlag, 2001, s. 251–261. Dostupné z URL: <http://link.springer.com/chapter/10.1007%2F3-540-44709-1_21>.

[11] TIMMERMAN T. R. *Mutual Information Analysis for Side Channel Attacks.* [MASTER'S THESIS], Eindhoven: Technische Universiteit Eindhoven, 2011, 128 s. Dostupné z URL: <http://alexandria.tue.nl/extra1/afstversl/wsk-i/timmerman2011.pdf>.

[12] GIERLICHS B., BATINA L., TUYLS P., PRENEEL B. *Mutual Information Analysis A Generic Side-Channel Distinguisher., CHES 2008, volume 5141 of Lecture Notes in Computer Science*, pages 426-442. Springer-Verlag, 2008.

[13] CHAUX A. *Study and implementation of the Mutual Information Analysis*, [MASTER'S THESIS], Bordeaux: University of Bordeaux, 2015, 49 s. Dostupné z URL: <http://mastercsi.labri.fr/wp-content/uploads/2015/10/CHAUX_Antoine-Memoire.pdf>.

[14] MEDWED M. *Overview of Countermeasures against Implementation Attacks*, prezentace, Albena, UCL Crypto Group, 2011, 40 s. Dostupné z URL: <https://www.cosic.esat.kuleuven.be/ecrypt/courses/albena11/slides/marcel_medwed_countermeasures.pd>.

[15] GOUBIN L. *A Sound Method for Switching between Boolean and Arithmetic Masking. In Proceedings of the Third International Workshop on Cryptographic Hardware and Embedded Systems*, London, UK: Springer-Verlag, 2001, s. 3-15, ISBN 3-540-42521-7. Dostupné z URL: <http://link.springer.com/chapter/10.1007%2F3-540-44709-1_2>.

[16] CORON J.-S., GOUBIN L. *On Boolean and Arithmetic Masking against DifferentialPower Analysis. In Proceedings of the Second International Workshop on Cryptographic Hardware and Embedded Systems*, London, UK: Springer-Verlag, 2000, s. 231-237, ISBN 3-540-41455-X. Dostupné z URL: <http://link.springer.com/chapter/10.1007%2F3-540-44499-8_18>.

[17] AKKAR M.-L., GIRAUD C. *An Implementation of DES and AES, Secure against Some Attacks. In Cryptographic Hardware and Embedded Systems*, Berlin / Heidelberg, 2001, s. 309-318, ISBN 978-3-540-42521-2. Dostupné z URL: <http://dx.doi.org/10.1007/3-540-44709-1_26

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

AES	Advanced Encryption Standard, šifrovací algoritmus
CPA	Correlation Power Analysis – Korelační proudová analýza
CvM	Cramer–von Mises, statistický test
DES	Data Encryption Standard, šifrovací algoritmus
DPA	Differential Power Analysis – Diferenciální proudová analýza
HW	Hamming Weight – Hammingova váha, proudový model
HD	Hamming Distance – Hammingova vzdálenost, proudový model
KS	Kolmogorov–Smirnof, statistický test
MIA	Mutual Information Analysis – Analýza vzájemnou informací
NIST	National Institute of Standards and Technology, normalizační institut USA
SPA	Simple Power Analysis – Jednoduchá proudová analýza
ZV	Zero-Value – Nulová hodnota, proudový model

SEZNAM PŘÍLOH

A ZDROJOVÉ KÓDY

A.1 CPA_MIA_compareCountTraces.m

```
% Skript pro zjisteni poctu proudovych prubehu nutnych k odhaleni ... tajneho
% klice. Samostatne nespustitelny. Pro zmenu mista (nejvyssiho vrcholu)
% vypoctu zmenit hodnotu promenne "int_start". Pro zmenu presnosti ... vypoctu
% zmenit hodnotu promenne "krok".

for metoda = 1:2

    if metoda == 1
        method = ('correlation');
        disp('Porovnavam - CA');
        int_start = 4266;
    else
        method = ('mutual');
        disp('Porovnavam - MIA');
        int_start = 4265;
    end

    int_end = int_start;
    traces_S = [];
    traces_S = [traces(:,int_start:int_end)];
    traces = traces_S;
    clear('traces_S');

    load('Tabulky_n.mat');
    krok=10;
    ulozitko = 1;

    for iteration = krok:krok:2000

        for main_klic = 1
            key_byte = main_klic;

            number_traces =iteration;
            model = HammingWeight;

            for num_traces_attack =number_traces

                for klic = key_byte
```

```
39                      input = [];
40                      trace = [];
41                      offset = [];
42                      for i = 1:krok:num_traces_attack
43                          input = [input;inputs(i:i+krok-1,klic)];
44                          input = double(input);
45                          trace = [trace;traces(i:i+krok-1,:)];
46                          trace = double(trace);
47                          offset = [offset;offsets(i:i+krok-1,klic)];
48
49                          for j =size(input,1)
50                              key = repmat((0:255),j,1);
51                              AddRound = bitxor(repmat(input,1,256),key);
52                              Mask2 = Masks(offset+2,1);
53                          end
54
55                          after_sbox = SubBytes(AddRound+1);
56                          after_sbox = ...
                               bitxor(after_sbox,repmat(Mask2,1,256));
57                          hw_xor_sbox_in_out = model(after_sbox+1);
58                          pre = trace;
59                      end
60                  end
61                  for i = 1:256
62                      switch method
63                          case 'correlation'
64                              corr_mat = corrcoef([pre  ...
                                   hw_xor_sbox_in_out(:,i)]);
65                              [m,n] = size(corr_mat);
66                              dpa_result(:,i) = corr_mat(m,1:n-1);
67                          case 'mutual'
68                              [m,n] = size(pre);
69                              for j=1:n
70                                  dpa_result(i,j)= ...
                                       mutualinfo(pre(:,j),hw_xor_sbox_in_out(:,i));
71                              end
72                      end
73                  end
74
75                  korelace_traces(ulozitko,:) = dpa_result;
76                  ulozitko = ulozitko +1;
77
78                  clear dpa_result;
79              end
80          end
81      end
```

```
82
83      figure('Visible','off');
84      plot(korelace_traces,'b');
85      hold on;
86      plot(korelace_traces(:,secret_new(1,1)+1),'r');
87      title(['Rozsynchronizovano ',num2str(hlavni_cyklus*5-5),'% ...
            prubehu, SNR = ',num2str(SNR),' dB.'], 'FontSize', 20);
88
89      xlabel('Pocet prubehu','FontSize',18,'FontWeight','bold');
90      popis_y = strcmp(method,'correlation');
91      if popis_y == 1
92          ylabel('Korelacni ...
                koeficient','FontSize',18,'FontWeight','bold')
93      else
94          ylabel('Vzajemna informace','FontSize',18,'FontWeight','bold')
95      end
96
97      nazevGrafu_rozptyl = num2str(hlavni_cyklus*5-5);
98      nazevGrafu_sum = num2str(SNR);
99
100     nazevGrafu = ...
            strcat(method,'-r',nazevGrafu_rozptyl,'snr',nazevGrafu_sum,'-compare-TEST');
101     saveas(gcf,nazevGrafu,'png');
102 end
```

A.2 CPA_MIA_error_gen.m

```
% Skript pro generovani chyb. Spustitelny. Promenne "int_start" a ... "int_end"
% urcuji oblast hledani nejvyssiho vrcholu. Promenna "rozptyl" udava
% maximalni posun pri desynchronizaci. Promenna "pocet_DS" definuje ... pocet
% datovych souboru pouzitych pri vypoctu. Pro nastaveni "idealnich"
% podminek nastavte promenne "SNR > 100" a "hlavni_cyklus = 1".

clc
clear all

int_start = 268000;
int_end = 273000;

rozdil = int_end - int_start;

pocet_DS = 2;
rozptyl = 20;

num_traces_testing = pocet_DS*1000; % Pocet testovanych prubehu.
err_lines = randperm(num_traces_testing); % Prehaze poradi radku.
err_vec = 0:5:100; % Vygeneruje cisla 0-100 s krokem 5, slouzici k ... urceni procent posunutych radku.
posun = randi([-rozptyl rozptyl],1,num_traces_testing); % Pro ... mnozstvi testovanych radku vytvori hodnotu jejich posunu.

traces_all = [];
inputs_S=[];
offsets_S=[];
shuffle0_S = [];
shuffle10_S = [];

disp('Nahravam prubehy.');
for krok = 1:pocet_DS
    nazev =strcat('DPAC_V42_k0_',num2str(krok),'000');
    load(nazev);
    indices = 1:1000;
    traces_all = [traces_all; ...
       traces(indices,int_start-rozptyl:int_end+rozptyl)];
    inputs_S=[inputs_S;inputs(indices,:)];
    offsets_S= [offsets_S;offsets(indices,:)];
    shuffle0_S = [shuffle0_S;shuffle0(indices,:)];
    shuffle10_S = [shuffle10_S;shuffle10(indices,:)];
```

```
39      clear('traces','inputs','offsets','shuffle0','shuffle10','indices')
40  end
41
42  inputs= inputs_S;
43  offsets=offsets_S;
44  shuffle0 = shuffle0_S;
45  shuffle10 = shuffle10_S;
46  clear('inputs_S','offsets_S','shuffle0_S','key','shuffle10_S','int_start','int_end'
47
48  disp('Ukladam.');
49  save('DPA_var.mat') % Ulozi vsechny konstanty, aby vypocty ...
        probihaly za stejnych podminek.
50
51  for SNR = 40:-8:8
52      fprintf(1,'SNR = %d dB.\n',SNR);
53      load('DPA_var.mat');
54
55      traces_all = double(traces_all);
56      for sumR = 1:num_traces_testing
57          traces_all(sumR,:) = ...
                awgn(traces_all(sumR,:),SNR,'measured','db');
58      end
59
60      traces_all = fix(traces_all);
61      traces_all = int8(traces_all);
62
63      for hlavni_cyklus = 1:21
64
65          fprintf(1,'Rozsynchronizovano %d%% ...
                prubehu\n',hlavni_cyklus*5-5);
66
67          asyn_lines = ...
                err_vec(hlavni_cyklus)*(num_traces_testing/100); % ...
                urceni mnozstvi rozsynchronizovanych radku
68          do_err = err_lines(1,1:asyn_lines); % Z vektoru err_lines ...
                vybere dane mnozstvi radku
69          do_err = sort(do_err); % Serazeni posouvanych radku
70
71          traces_S = [];
72          traces_S = traces_all(:,rozptyl+1:rozdil+rozptyl+1);
73
74          for i = 1:asyn_lines
75              err_line_1st = traces_all(do_err(1,i),:); % Nacte ...
                    radek, ktery ma chybovat
76              err_line_2nd = err_line_1st(1,1 + rozptyl + ...
                    posun(1,i):rozdil + 1 + rozptyl + posun(1,i)); % ...
```

```
                Posune zacatek radku o hodnotu danou konstantou posun
77              traces_S(do_err(1,i),:) = err_line_2nd; % Do traces_S ...
                    zapise upraveny radek.
78          end
79
80          figure('Name','Namerene ...
                prubehy','NumberTitle','off','Visible','off');
81          plot(traces_S')
82          title(['Rozsynchronizovano ',num2str(hlavni_cyklus*5-5),'% ...
                prubehu, SNR = ',num2str(SNR),' dB.'], 'FontSize', 20);
83
84          axis([0 5000 -150 150]);
85          xlabel('Cas','FontSize',18,'FontWeight','bold');
86          ylabel('Amplituda','FontSize',18,'FontWeight','bold')
87
88          nazevGrafu_rozptyl = num2str(hlavni_cyklus*5-5);
89          nazevGrafu_sum = num2str(SNR);
90
91          nazevGrafu = ...
                strcat('input-r',nazevGrafu_rozptyl,'snr',nazevGrafu_sum,'-TEST');
92          saveas(gcf,nazevGrafu,'png');
93
94          traces = traces_S;
95
96          CPA_MIA_uknownKey
97
98          CPA_MIA_compareCountTraces
99
100         clearvars -except SNR hlavni_cyklus err_lines err_vec ...
                inputs num_traces_testing offsets posun rozdil rozptyl ...
                shuffle0 shuffle10 traces_all;
101     end
102 end
```

A.3 CPA_MIA_uknownKey.m

```
% Skript pro hledani tajneho klice. Samostatne nespustitelny. Pro zmenu
% presnosti vypoctu zmenit hodnotu promenne "krok".

for metoda = 1:2

    if metoda == 1
        method = ('correlation');
        disp('Porovnavam - CA');
    else
        method = ('mutual');
        disp('Porovnavam - MIA');
    end

    load('Tabulky_n.mat');
    krok= 10;

    for main_klic = 1
        key_byte = main_klic;

        number_traces = length(traces(:,1));
        temp = mod(number_traces,krok);
        number_traces = number_traces-temp;
        clear temp
        model = HammingWeight;

        for num_traces_attack =number_traces

            for klic = key_byte
                input = [];
                trace = [];
                offset = [];
                for i = 1:krok:num_traces_attack
                    input = [input;inputs(i:i+krok-1,klic)];
                    input = double(input);
                    trace = [trace;traces(i:i+krok-1,:)];
                    trace = double(trace);
                    offset = [offset;offsets(i:i+krok-1,klic)];

                    for j =size(input,1)
                        key = repmat((0:255),j,1);
                        AddRound = bitxor(repmat(input,1,256),key);
                        Mask2 = Masks(offset+2,1);
                    end
```

```
44
45                      after_sbox = SubBytes(AddRound+1);
46                      after_sbox = ...
                           bitxor(after_sbox,repmat(Mask2,1,256));
47                      hw_xor_sbox_in_out = model(after_sbox+1);
48                      pre = trace;
49                  end
50              end
51
52              for i = 1:256
53                  switch method
54                      case 'correlation'
55                          corr_mat = corrcoef([pre  ...
                               hw_xor_sbox_in_out(:,i)]);
56                          [m,n] = size(corr_mat);
57                          dpa_result(i,:) = corr_mat(1:n-1,m);
58                      case 'mutual'
59                          [m,n] = size(pre);
60                          fprintf(1,'key %d\n',i);
61                          for j=1:n
62                              dpa_result(i,j)= ...
                                   mutualinfo(pre(:,j),hw_xor_sbox_in_out(:,i));
63                          end
64                  end
65              end
66
67              CPA = dpa_result;
68              CPA_c(klic,:) = dpa_result(secret_new(1,klic)+1,:);
69
70              clear dpa_result;
71          end
72      end
73
74      if metoda == 1
75          graf_medCPA = CPA;
76      else
77          medCPA=median(CPA);
78          [y,x] = size(CPA);
79          for i = 1:x
80              graf_medCPA(:,i)=CPA(:,i)-medCPA(i);
81          end
82      end
83
84      figure('Visible','off');
85      plot(transpose(graf_medCPA),'b')
86      hold on;
```

```
87      plot(graf_medCPA(secret_new(1,1)+1,:),'r')
88      axis([0 5000 -2 2]);
89      title(['Rozsynchronizovano ',num2str(hlavni_cyklus*5-5),'% ...
            prubehu, SNR = ',num2str(SNR),' dB.'], 'FontSize', 20);
90      nazevGrafu_rozptyl = num2str(hlavni_cyklus*5-5);
91      nazevGrafu_sum = num2str(SNR);
92
93      xlabel('Cas','FontSize',18,'FontWeight','bold');
94      popis_y = strcmp(method,'correlation');
95      if popis_y == 1
96          ylabel('Korelacni ...
                koeficient','FontSize',18,'FontWeight','bold')
97      else
98          ylabel('Vzajemna informace','FontSize',18,'FontWeight','bold')
99      end
100
101     nazevGrafu = ...
            strcat(method,'-r',nazevGrafu_rozptyl,'snr',nazevGrafu_sum,'-peak-TEST');
102     saveas(gcf,nazevGrafu,'png');
103 end
```

A.4 dpa_mia.m

```
1     disp('Predicting the instantaneous power consumption ...');
2     power_consumption = bitget(after_sbox,1);
3
4     % correlate the predicted power consumption with the real power
5     % consumption
6     disp('Generating the difference traces ...');
7
8     for i=first:last
9         key_trace(i,:) = ...
              sum(traces(find(power_consumption(:,i)==1),:)) - ...
              sum(traces(find(power_consumption(:,i)==0),:));
10    end
11 end
12
13 if (strcmp(method,'correlation'))
14
15    % predict the power consumption
16
17    disp('Predicting the instantaneous power consumption ...');
18    power_consumption = byte_Hamming_weight(after_sbox+1);
19
20    % correlate the predicted power consumption with the real power
21    % consumption
22    disp('Generating the correlation traces ...');
23
24    chunksize=50;
25    chunks=n/50;
26    for i=first:last
27        for j=1:chunks
28            cmatrix= ...
                  corrcoef([traces(:,1+(j-1)*chunksize:j*chunksize)  ...
                  power_consumption(:,i)]);
29            key_trace(i,1+(j-1)*chunksize:j*chunksize) ...
                  =cmatrix(chunksize+1,1:chunksize);
30        end
31    end
32 end
33
34 if (strcmp(method,'mutual'))
35
36    % predict the power consumption
37
38    disp('Predicting the instantaneous power consumption ...');
```

```
39      power_consumption = byte_Hamming_weight(after_sbox+1);
40
41      % correlate the predicted power consumption with the real power
42      % consumption
43      disp('Generating traces ...');
44
45      parfor i=1:256
46          for j=1:n
47               key_trace(i,j)= ...
                     mutualinfo(traces(:,j),power_consumption(:,i));
48          end
49      end
50  end
51
52  % Pro Korelacni proudovou analyzu a Analyzu vzajemnou informaci, ...
       lze pro
53  % pouzit vykresleni prubehu pouzit nize uvedenou cast. U Kocherovy ...
       metody
54  % zobrazi spatny klic. Je to zpusobeno ti, ze se nehleda nejvyssi mozna
55  % spicka, ale nejvetsi rozdil mezi hodnotami ve vystupnich grafech.
56  % Nejlepsi mozna metoda zobrazeni je pouziti skriptu "show_plots.m" ze
57  % stranek z dpabook.org.
58
59  [value, location] = max(key_trace(:));
60  [R,C] = ind2sub(size(key_trace),location);
61  figure
62  plot(key_trace(R,:),'b');
63  title(['Graf hodnot vzajemne informace pro klic ',num2str(R-1)]);
64  disp(['Hodnota klice ',num2str(b),'. bitu je ',num2str(R-1),'.'])
65  xlabel('Cas');
66  popis_y = strcmp(method,'correlation');
67  if popis_y == 1
68      ylabel('Korelacni koeficient')
69  else
70      ylabel('Vzajemna informace')
71  end
```

A.5 dpa_mia_porovnani.m

```
% Na pocatku zvolte metodu vypoctu Dale je mozne volit hodnoty ... promennych
% "first" pro prvni hledany klic a "last" pro posledni. V pracovnim
% prostoru WS1.mat jsou vstupni hodnoty pouze pro prvni bit klice. Nema
% tedy vyznam menit hodnoty promenne "b" oznacujici byte ... podrobovany utoku.
% Ze stranek dpabook.org je mozne stahnout jeste workspace WS2.mat, ... ktery
% bude v tomto skriptu take fungovat. Navic bude mozne volit i byte na
% ktery bude veden utok.

clc
clear all

workspace = ('WS1.mat');
first = 1;
last = 256;

% correlation, mutual
method = ('correlation');

load('-mat',workspace);
b=1;
inputs=inputs(:,b);
key = [0:255];

step=2;

max_peak = 4743;
traces = traces(:,max_peak);
counter = 1;

for iteration = step:step:200;
        for attacked_traces = iteration
            input = [];
            trace = [];
            for a = 1:step:attacked_traces
                fprintf(1,'traces %d - %d\n',a,a+step-1);
                input = [input;inputs(a:a+step-1,:)];
                trace = [trace;traces(a:a+step-1,:)];

                for i=1:size(input,1)
                    after_sbox(i,:) = SubBytes(bitxor(input(i),key)+1);
```

```
41                  end
42                  power_consumption = byte_Hamming_weight(after_sbox+1);
43              end
44              for i=first:last
45                  if (strcmp(method,'correlation'))
46                      corr_mat = corrcoef([trace  ...
                            power_consumption(:,i)]);
47                      key_trace(:,i) = corr_mat(2,1);
48                  elseif (strcmp(method,'mutual'))
49                      mutu_mat = mi(trace,power_consumption(:,i));
50                      key_trace(:,i) = mutu_mat;
51                  end
52              end
53
54              result(counter,:) = key_trace;
55              counter = counter +1;
56              clear key_trace;
57          end
58  end
59
60  plot(result)
61  xlabel('Pocet prubehu','FontSize',18,'FontWeight','bold');
62  popis_y = strcmp(method,'correlation');
63  if popis_y == 1
64      ylabel('Korelacni koeficient','FontSize',18,'FontWeight','bold')
65  else
66      ylabel('Vzajemna informace','FontSize',18,'FontWeight','bold')
67  end
```

A.6 mutualinfo.m

```
function z = mutualInformation(x, y)
% Compute mutual information I(x,y) of two discrete variables x and y.
% Written by Mo Chen (mochen80@gmail.com).
    assert(numel(x) == numel(y));
    n = numel(x);
    x = reshape(x,1,n);
    y = reshape(y,1,n);

    l = min(min(x),min(y));
    x = x-l+1;
    y = y-l+1;
    k = max(max(x),max(y));

    idx = 1:n;
    Mx = sparse(idx,x,1,n,k,n);
    My = sparse(idx,y,1,n,k,n);
    Pxy = nonzeros(Mx'*My/n); %joint distribution of x and y
    Hxy = -dot(Pxy,log2(Pxy+eps));

    Px = mean(Mx,1);
    Py = mean(My,1);

    % entropy of Py and Px
    Hx = -dot(Px,log2(Px+eps));
    Hy = -dot(Py,log2(Py+eps));

    % mutual information
    z = Hx + Hy - Hxy;
%endfunction
```

B OBSAH PŘILOŽENÉHO DVD

- Složka „Skripty“ - Obsahuje všechny skripty uvedené v práci. Byly testovány v MATLAB R2015a.
  - `CPA_MIA_compareCountTraces.m` - skript pro zjištění počtu proudových průběhů, nutných k odhalení tajného klíče, samostatně nespustitelný. (DPA Contest 4.2)
  - `CPA_MIA_error_gen.m` - skript pro generování chyb, spustitelný. (DPA Contest 4.2)
  - `CPA_MIA_uknownKey.m` - skript pro hledání tajného klíče, samostatně nespustitelný. (DPA Contest 4.2)
  - `dpa_mia.m` - skript pro hledání tajného klíče, samostatně spustitelný. (DPAbook.org)
  - `dpa_mia_porovnani.m` - skript pro zjištění počtu proudových průběhů, nutných k odhalení tajného klíče, samostatně spustitelný. (DPAbook.org)
  - `mutualinfo.m` - funkce pro výpočet vzájemné informace, samostatně spustitelná.
- Složka „Text“ - Elektronická verze diplomová práce
  - Složka „LaTEX“ - Elektronická verze zpracovaná v LaTexu.
  - Složka „PDF“ - Elektronická verze v .pdf
- Složka „Workspaces“ - Obsahuje soubory typu .mat obsahující konstanty a počáteční hodnoty.
  - `DPA_var.mat` - soubor vstupních hodnot, použitých v této práci. Pokud bude ve skriptu `CPA_MIA_error_gen.m` prvních 43 řádků a bude použit tento vstupní soubor, bude možné reprodukovat všechna data uvedená v této práci.
  - `DPAC_V42_k0_X000.mat` - obsahují hodnoty vstupních měření pro kapitolu DPA Contest 4.2.
  - `Tabulky_n.mat` - doplňující konstanty a proměnné pro DPA Contest 4.2.
  - `WS1.mat` - obsahuje hodnoty vstupních měření pro kapitolu DPABook.